BBN Systems and Technologies Corporation

A Subsidiary of Bolt Beranek and Newman Inc.

Report No. 7143

IRUS-II User Manual

Dawn MacLaughlin, Kimberle Koile, and Edward Walker

September 1989

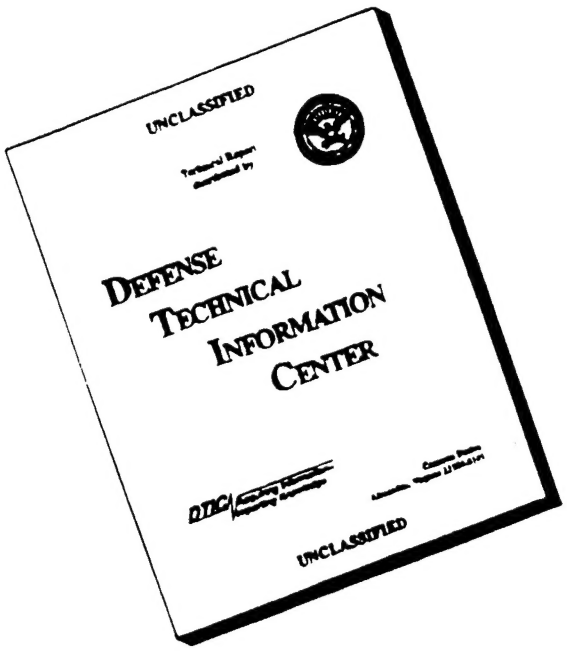
Prepared by:

BBN Systems and Technologies Corporation
10 Moulton Street
Cambridge, MA 02138

Prepared for:

Defense Advanced Research Projects Agency
1400 Wilson Blvd.
Arlington, VA 22209

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by ONR under Contract No. N00014-85-C-0016. The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT APPROVED FOR PUBLIC RELEASE Unlimited DISTRIBUTION UNLIMITED
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) 7143	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION BBN Systems and Technologies 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research
6c. ADDRESS (City, State, and ZIP Code) 10 Moulton Street Cambridge, MA 02138	7b. ADDRESS (City, State, and ZIP Code) Department of the Navy Arlington, VA 22217
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Defense Advanced Research Projects Agency 8b. OFFICE SYMBOL (If applicable) DARPA ISTO	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-85-C-0016
8c. ADDRESS (City, State, and ZIP Code) 1400 Wilson Blvd. Arlington, BA 22209	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

Irus-II user Manual

12. PERSONAL AUTHOR(S)
Dawn MacLaughlin, Kimberle Koile, and Edward Walker

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Interim technical Report	FROM _____ TO _____	1989, Semptember	61

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Janus, Irus, Natural Language Interfaces, User Interfaces

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This user's manual for the Irus-II natural language system describes the capabilities of the different modes of the end user interface for Irus-II, including entering and editing queries, the history list, control of processing, viewing of the intermediate results, window management and the use of files of queries. There is a companion document to this one, A Guide to Irus-II Application Development, BBN Report #7144, intended for use by developers, that describes the internal operation of Irus-II, including the dictionary, domain model, semantic interpretation rules, and back-end translation rules. Irus-II is part of a larger system called Janus, which is composed of a number of seamless interface utilities, uncluding understanding (provided by Irus-II), generation (provided by Spokesman), maps, and other graphics tools.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD Form 1473, JUN 86 Previous editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE

Table of Contents

List of Figures

1 Introduction

This user's manual for the IRUS-II natural language system describes the capabilities of the different modes of the end user interface for IRUS-II, including entering and editing queries, the history list, control of processing, viewing of intermediate results, window management, and the use of files of queries. There is a companion document to this one, **A Guide to IRUS-II Application Development**, BBN Report #7144, intended for use by developers, that describes more of the internal operation of IRUS-II, including the dictionary, domain model, semantic interpretation rules, and back-end translation rules. IRUS-II is part of a larger system called JANUS, which is composed of a number of seamless interface facilities, including understanding (provided by IRUS-II), generation (provided by Spokesman), maps, and other graphics tools. Note that the screen shots in this document were taken from this larger system and are labeled as such.

The different applications of IRUS require that the interface provide configurations for three different classes of users: 1) an end-user[1] configuration that supports easy and flexible input, editing, and processing of typed or mouse-selected queries and commands; 2) developer configurations for natural language experts who are developing, testing, debugging, or updating IRUS; 3) pop up windows for query-response interactions that may be used in conjunction with other interfaces. The available IRUS interface configurations use the Symbolics Lisp Machine window system to provide multiple pane displays and mouse and keyboard input. The interface can be developed from the generic capabilities described and illustrated below and tailored to the needs of a particular application.

2 Components of the IRUS Interface

Three basic panes form the building blocks of the interface. The relative sizes and visibility of panes, the general appearance of the screen, the contents of menus, etc., are readily modifiable to suit the particular end-user community or application environment of IRUS.

1. An **Editing pane**, which has Zmacs[2] entry and editing capabilities and mouse selectable commands, supports entering and processing queries; queries can be typed, selected by mouse, or entered from a file of stored queries.
2. A **History pane** maintains a list of successive queries and, where appropriate, responses. Queries in the History pane are mouse selectable, and pop-up menus permit selection of further actions on the selected item.
3. An **Interaction pane** provides detailed information concerning the results of each stage of IRUS processing. Entries in the Interaction pane are also mouse selectable, and a variety of pop-up menus permit selection of further actions on selected items.

---

[1]The term "end-user" refers to someone who is not involved with IRUS system development.

[2]Zmacs is the editor provided on the Symbolics Lisp Machine.

A typical end-user's interface for IRUS would consist of an editing pane with a subset of the available commands for query entry, editing, and processing or pop-up windows for interactions and viewing the history. A typical developer's interface would contain a configuration with all three types of panes to provide easy entry of test queries, checking of responses for related queries, and development and testing of IRUS internal structures such as the World Model Language (WML) and the target language code generator.

IRUS acquisition configurations for adding new words and new domain interpretation rules are provided for the developer and contain simple acquisition command menus and prompt and respond dialogs. Tools for domain model acquisition are provided by the knowledge representation system used by IRUS.

Depending on the nature of the application environment and of the items to be entered, these facilities might be adapted to provide system maintainers or even end-users with a means of expanding, maintaining, or updating the system.

2.1 Editing Pane

The Editing pane is used in the IRUS interface as a query input editor. The full Zmacs text editing facility of the Symbolics machine is available to the user for composing and editing queries. In addition to efficient text entry and editing, this editor supports macros, abbreviations, and other sophisticated editing features, which can be used to provide an efficient and customizable query entry system. The Editing pane also provides for aggregating queries and for writing queries into or reading queries from a file.

Mouse selectable **Edit**, **Process**, **Interrupt**, and **Other** commands have been provided.[3] These commands can also be invoked with keystrokes. A help facility, available by selecting the **Help** command or by pressing the **Help** key, has been provided. Help messages, as well as specialized query composition aids, can be added in response to user evaluation. For example, a command menu of simple editing commands might be provided to aid users who do not know which keystrokes are used for Zmacs editing functions, or a brief explanation of the types of acceptable queries might be incorporated into the help feature.

The developer's Editing pane can be used not only for editing English queries, but also for editing the Lisp expressions produced by intermediate stages of IRUS processing. Full Zmacs text and Lisp editing functions are available, depending on the editing mode, which is either set automatically by the system or selected by the user. In addition to allowing a choice of editing modes, the developer's Editing pane contains commands that allow selective break points in the processing of a query. The developer may choose to halt processing at any of several stages of IRUS processing. This feature makes all stages of processing available for examination and editing, and it is especially useful for debugging and experimental development of the individual components of IRUS.

[3]Unless otherwise stated, it is assumed that commands are selected by placing the mouse cursor over the command name and clicking the left mouse button.

2.2 History Pane

The scrollable History pane accumulates queries and responses. Queries in this window are selectable for editing, reprocessing, or for requesting a more detailed description in the Interaction pane. This reentry feature can be used to provide a convenient standard form for a sequence of similar queries.

2.3 Interaction Pane

The Interaction pane provides a mechanism for detailed display of the queries and Lisp expressions that result from stages of IRUS processing. Items in this pane may be selected for more detailed examination or for further editing. For example, the results of parsing and semantic interpretation may be displayed and edited, then submitted for further processing to test alternative versions of initial query processing.

3 Example Interface Configurations

The example screens in Figures 1, 2, and 3 show typical end-user and developer configurations. The end-user configuration appears when the interface is entered initially with a **<select> J**. It consists of a query editor, history, and output window, and a small set of commands allowing the user to edit or process a query, or to interrupt the processing at intermediate steps (see Figure 1).

Another end-user's version of the interface contains an Editing pane and a History pane in separate pop-up windows (see Figure 2). The pop-up History pane is optional and only appears when the user requests it (by selecting the **History** command). The location on the screen and the size of the pop-up windows are arbitrary and can be modified to reflect the available screen space or the needs of the user.

The developer's configuration illustrated here occupies the full screen and contains Editing, History, and Interaction panes (see Figure 3). An added command menu is provided to allow the developer to interrupt processing of a query at intermediate stages. The Interaction pane is used for detailed display of queries and intermediate processing steps. This pane is also a Common Lisp window, and the developer can type Lisp expressions in the pane and have them executed as if typing in a Symbolics provided Common Lisp window.

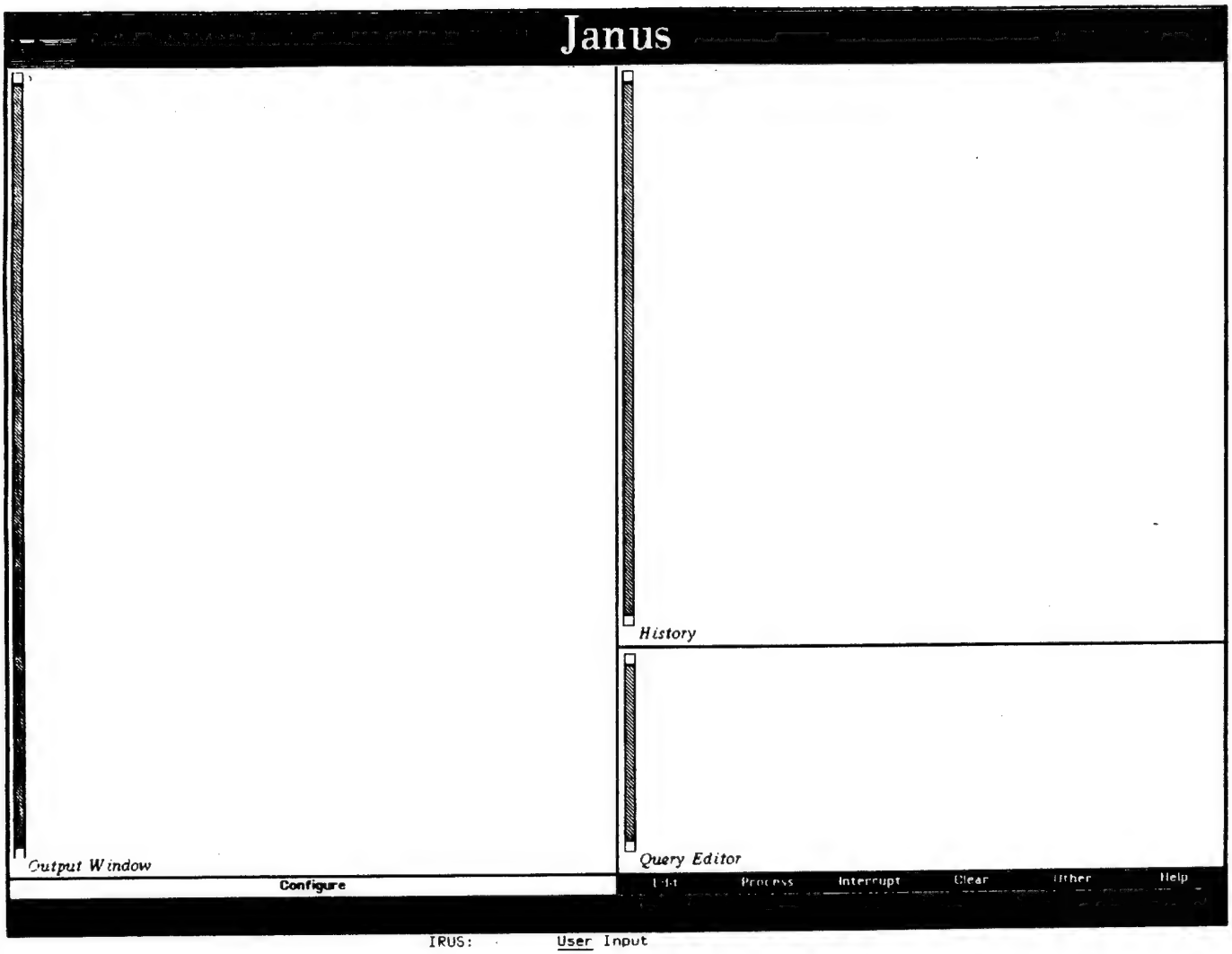


Figure 1: Typical end-user configuration

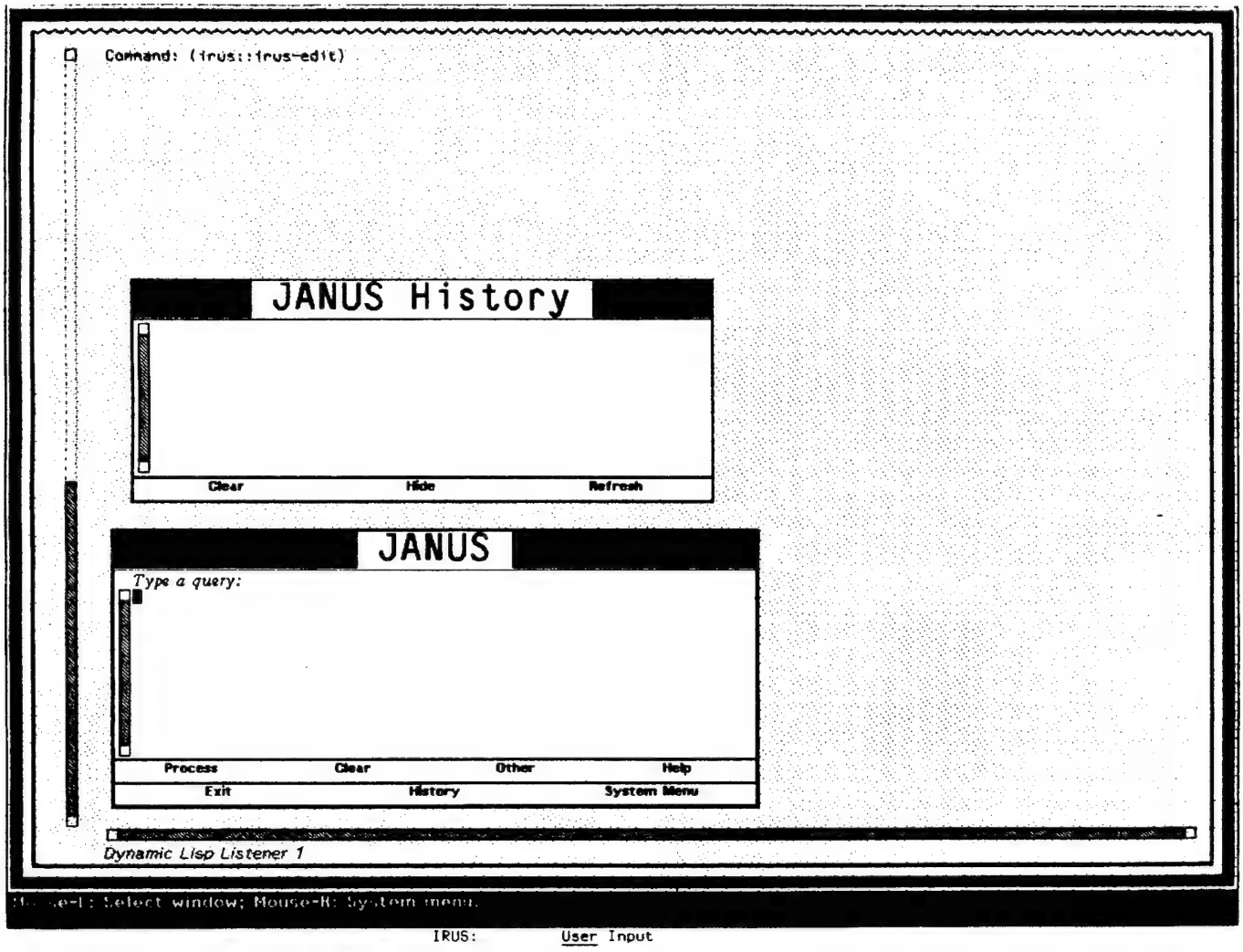


Figure 2: Typical pop-up window configuration

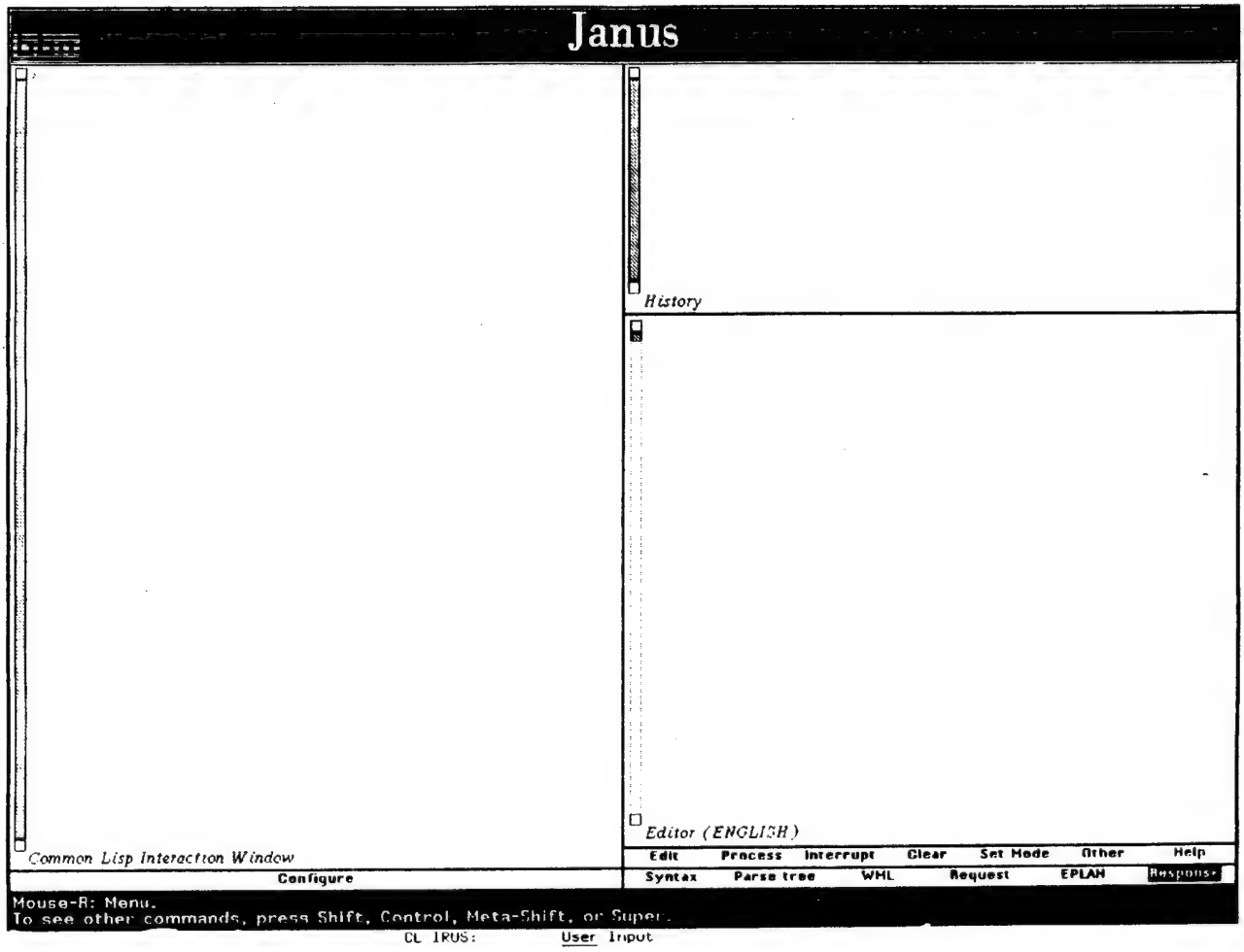


Figure 3: Typical developer configuration

4 Interaction with IRUS

The following scenarios illustrate typical interactions with IRUS.

4.1 End-User Interaction

In the first scenario, an end-user has selected the IRUS pop-up window (from a system menu, for example). A query is typed into the Editing pane using the keyboard and processed by pressing the **End** key or selecting the **Process** command at the bottom of the window. As processing takes place, the current stage of processing is printed in the system status line at the bottom of the full Symbolics screen in order to provide user feedback. (See Figure 4.)

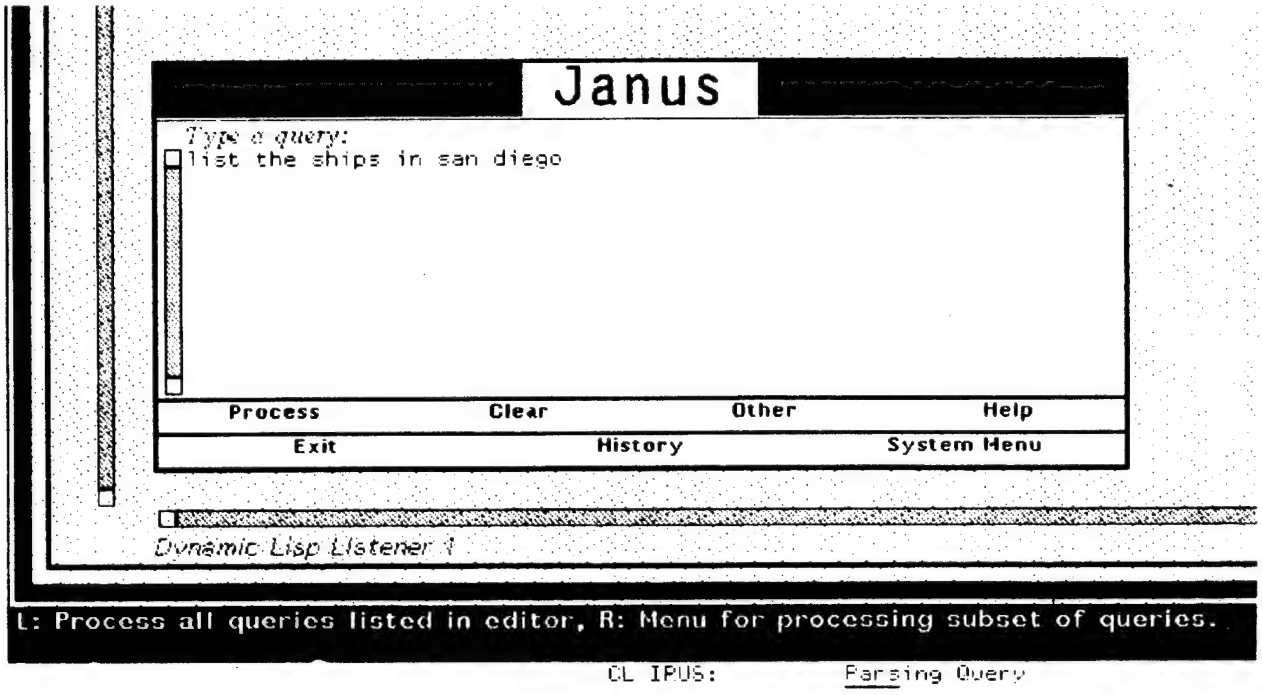


Figure 4: Query typed and **Process** command selected

Upon the completion of IRUS processing, the IRUS pop-up window disappears, a query object is returned, and the application program (e.g., FRESH) takes over command processing and response handling. These query objects contain the query string, the results of stages of IRUS processing, and input data needed for the application program.

Messages to the end-user are shown in a window that scrolls down over the editor pane. If, for example, IRUS processing does not succeed, IRUS attempts a telegraphic parse and prints "Trying for telegraphic parse ..." in

the scroll down window (see Figure 5). Typing a space causes the scroll down window to disappear. If the telegraphic parse fails, IRUS prints a failure message, leaving the user in input mode, ready to enter another query, press the help key, etc. The message facility can be used to notify users of failure and provide users with help messages for correcting errors of query form or content.

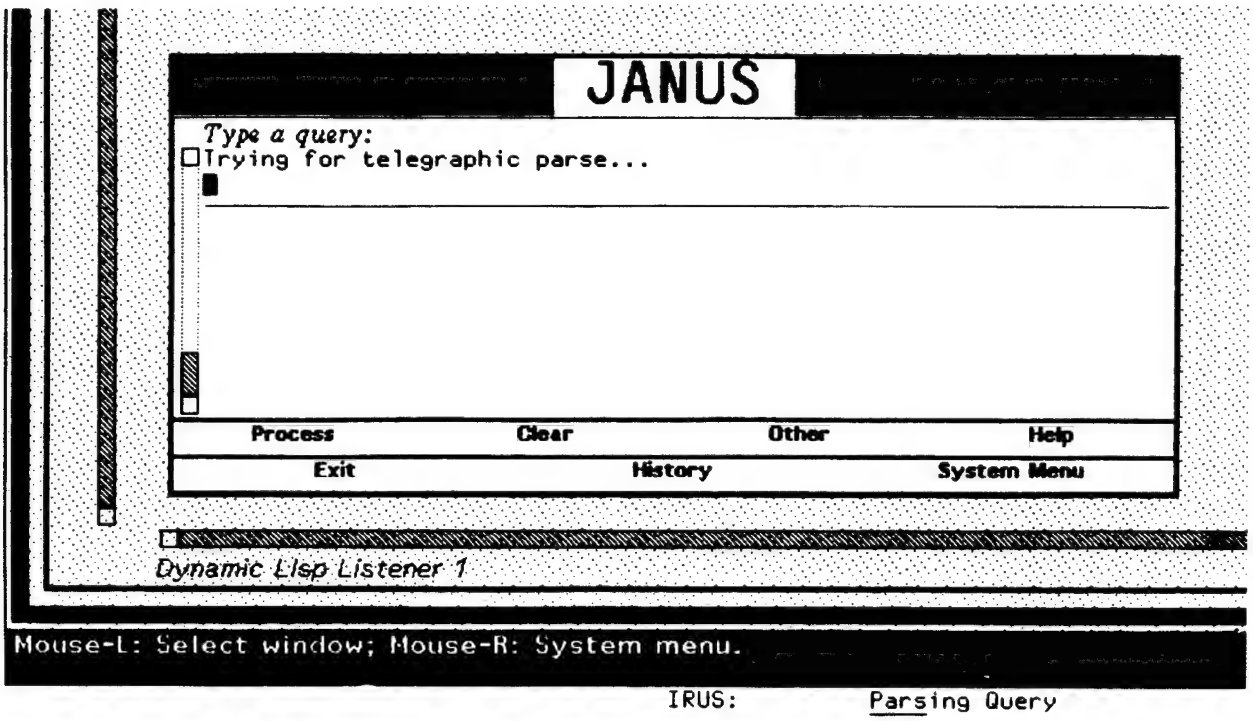


Figure 5: Message to end-user

Queries may be aggregated in the Editing pane by separating them with 2 carriage returns. All queries will be processed in sequence after the **End** key is pressed or the **Process** command is selected. Queries can be written from the Editing pane into a file or retrieved from a file and inserted into the Editing pane by selecting the **Other** command, which pops up a menu, then selecting **Write** or **Insert**, respectively (see Figures 6 through 11). This facility is useful for creating a set of queries that are processed on a regular basis.

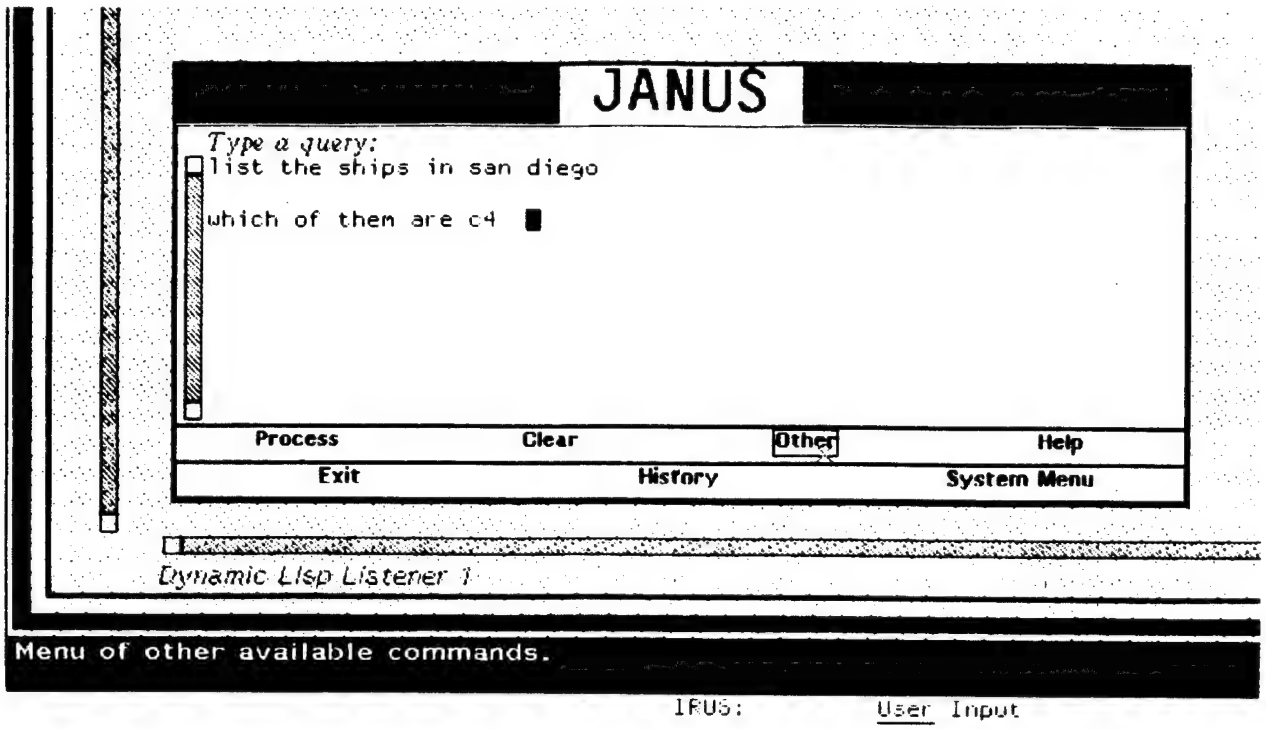


Figure 6: Select **Other** command

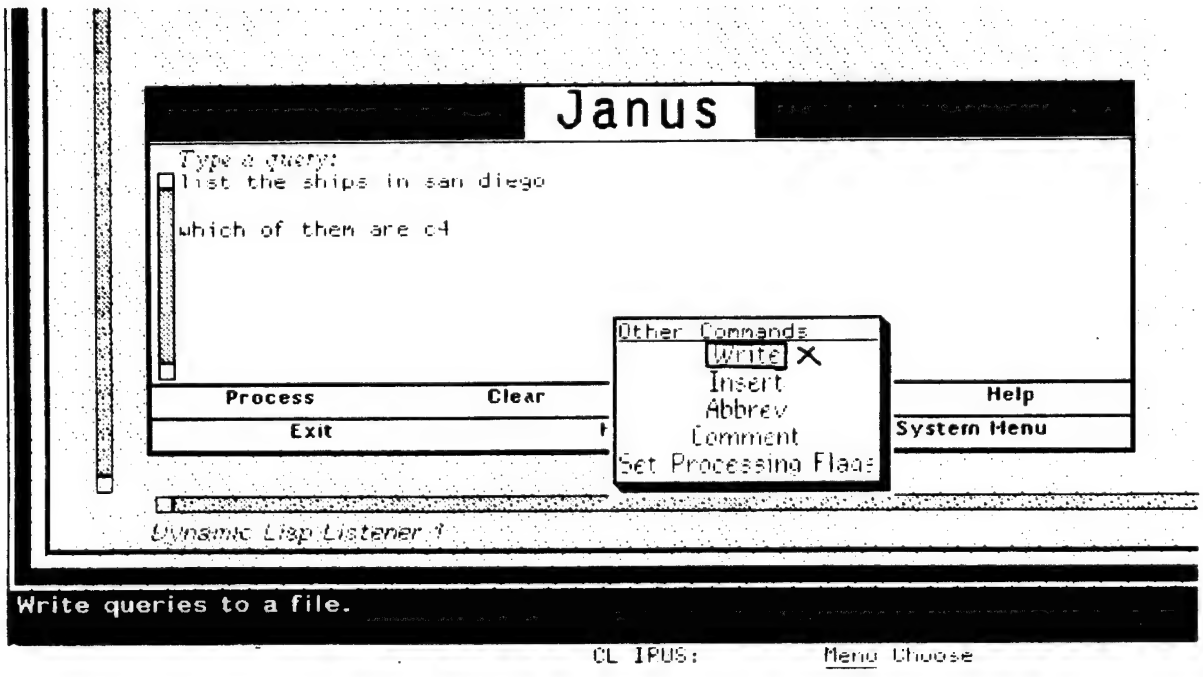


Figure 7: Select **Write** command

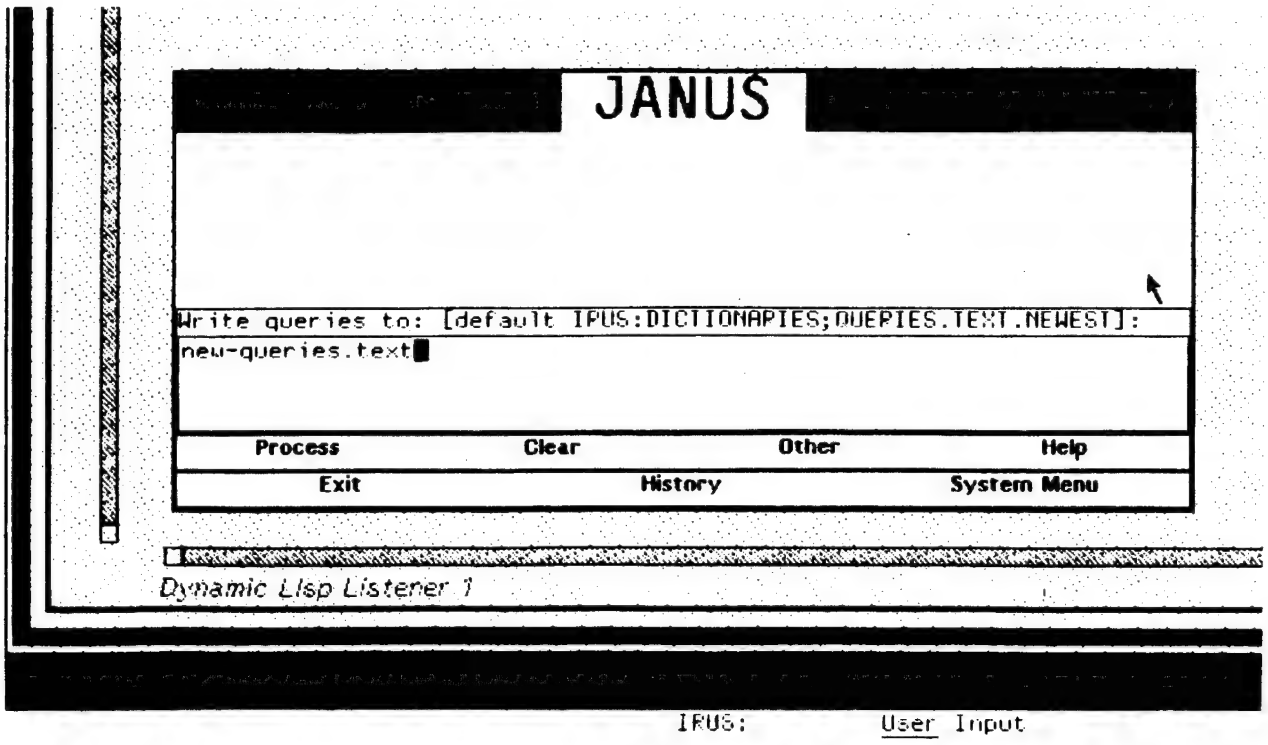


Figure 8: Prompt for query file name

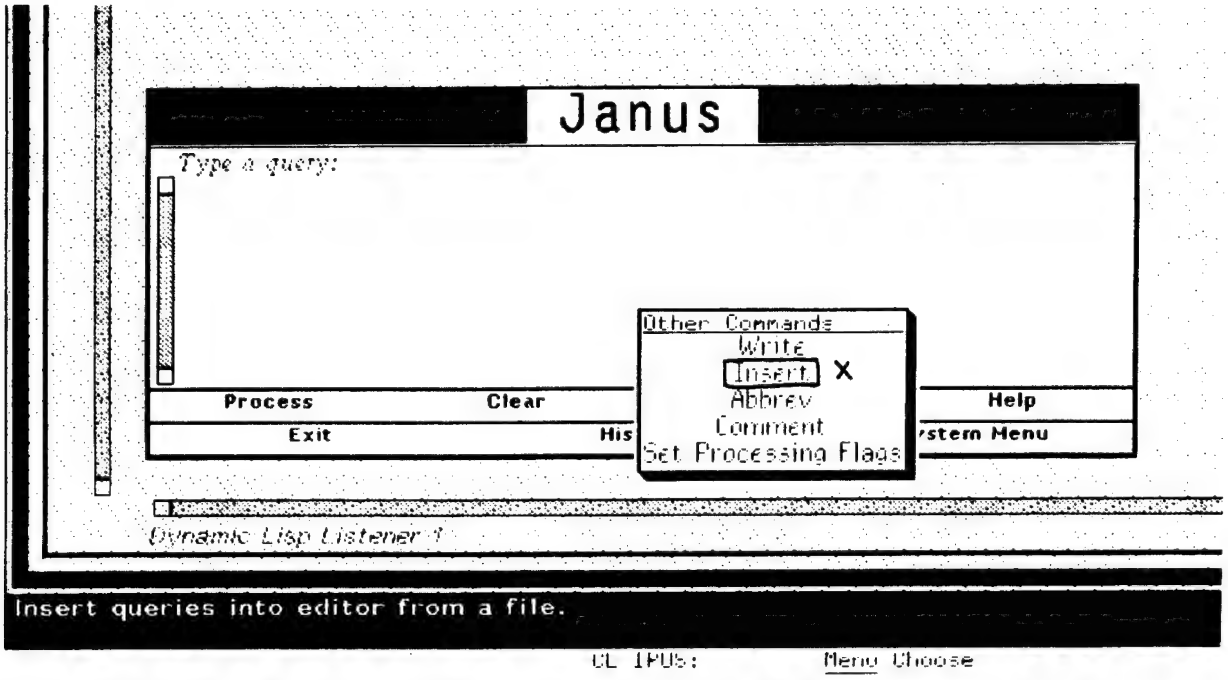


Figure 9: Select **Insert** command

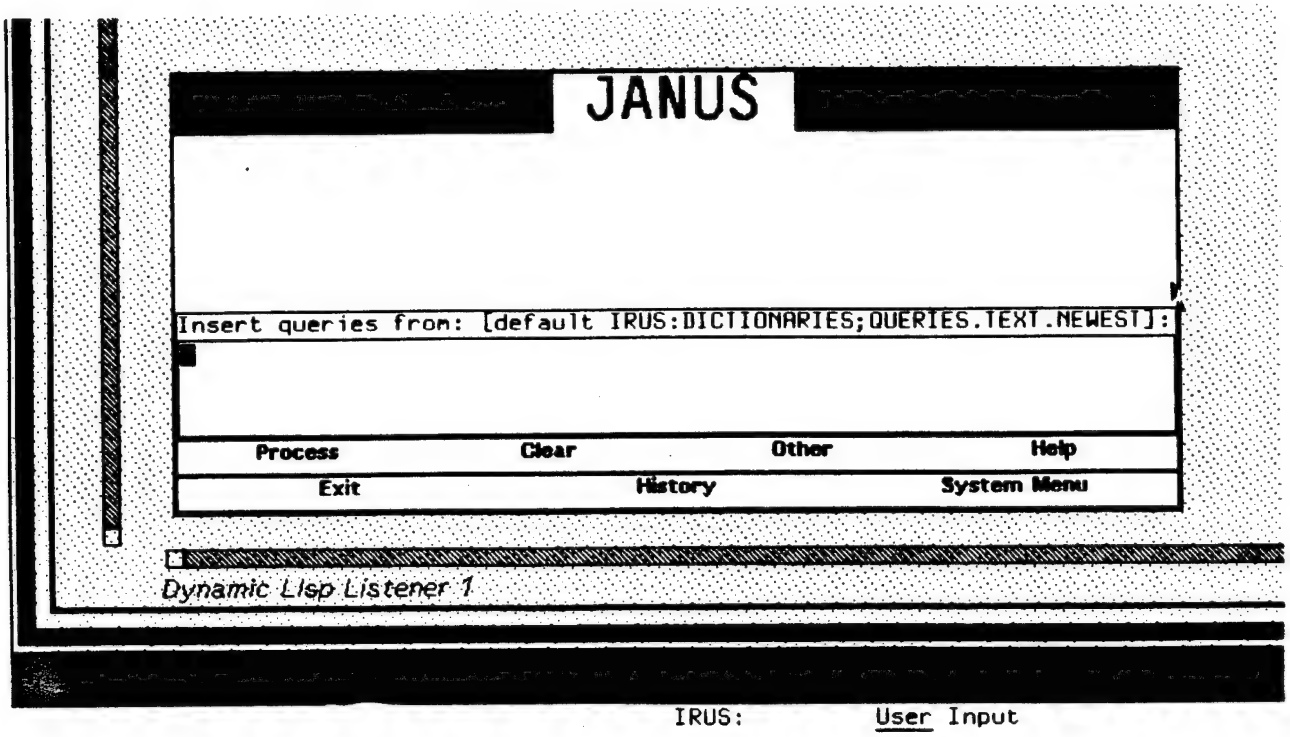


Figure 10: Prompt for query file name

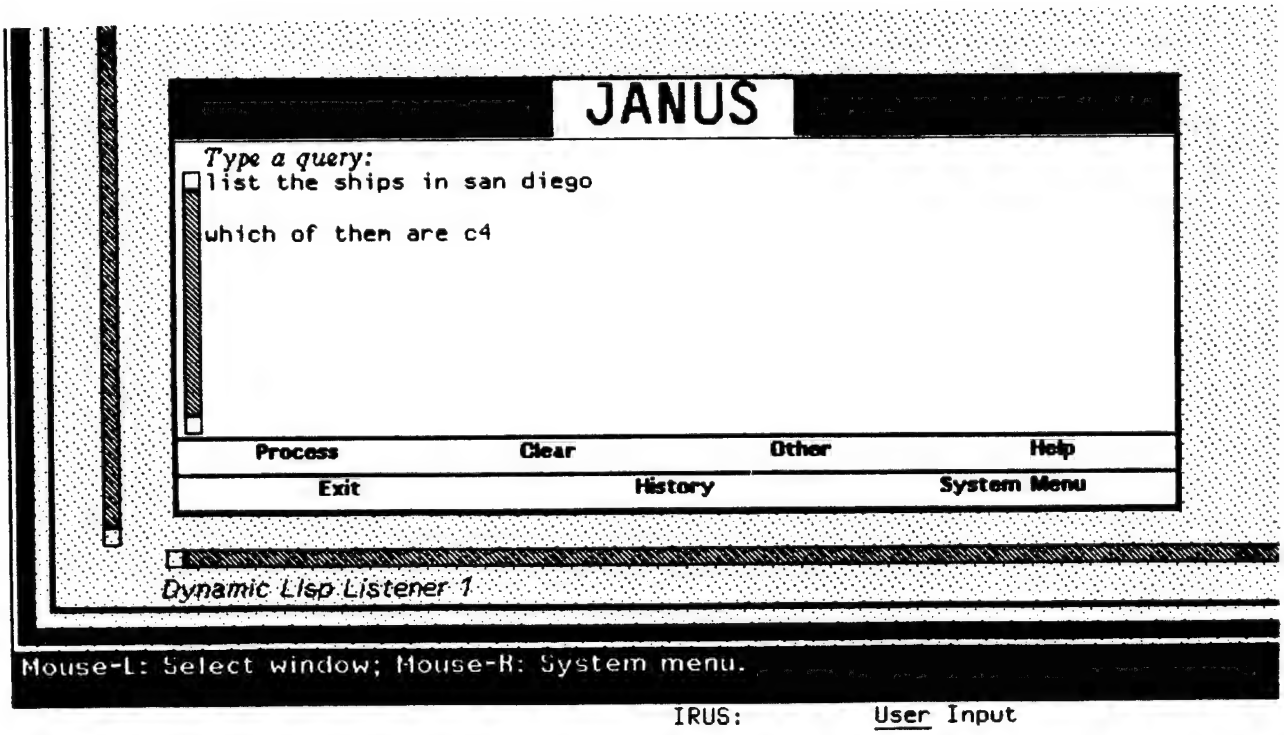


Figure 11: Queries inserted from file

The end-user may want to selectively process queries from a group of queries in the Editing pane. To do this, he or she marks queries by preceding them with a "*", clicking right on the **Process** command, and selecting one of the choices from the resulting pop-up menu: process all queries, only marked queries, or only unmarked queries (see Figure 12).

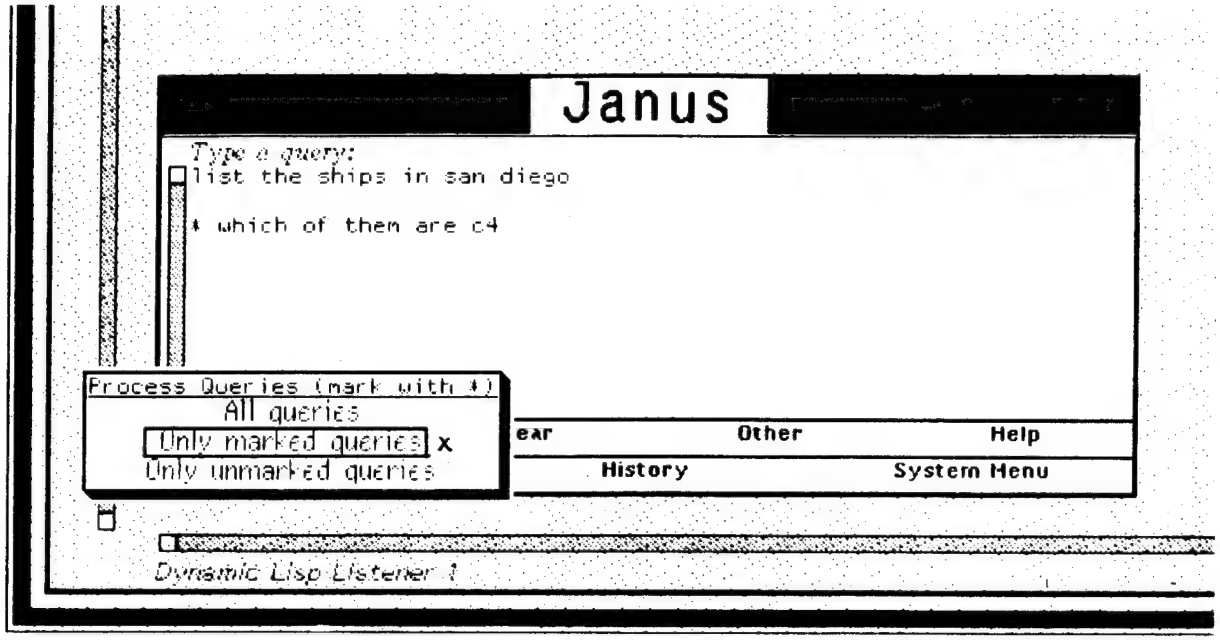


Figure 12: Selective processing of queries

The full Zmacs text editing facility of the Symbolics machine is available to the end-user for composing and editing queries. In addition to efficient text entry and editing, this editor supports macros, abbreviations, and other editing features, which can be used to provide an efficient and customizable query entry system. Thus, the port name "San Diego" (or even a whole query) might have been entered by typing only an abbreviation such as "SD", which is automatically expanded by the Zmacs editor, either by reference to a user customizable abbreviation file or to a system abbreviation file made available to all users. To create an abbreviation, the user would select the **Abbrev** command from the **Other** command menu, then select **Make Word Abbrev** from the abbreviation pop-up menu (see Figures 13 through 16).

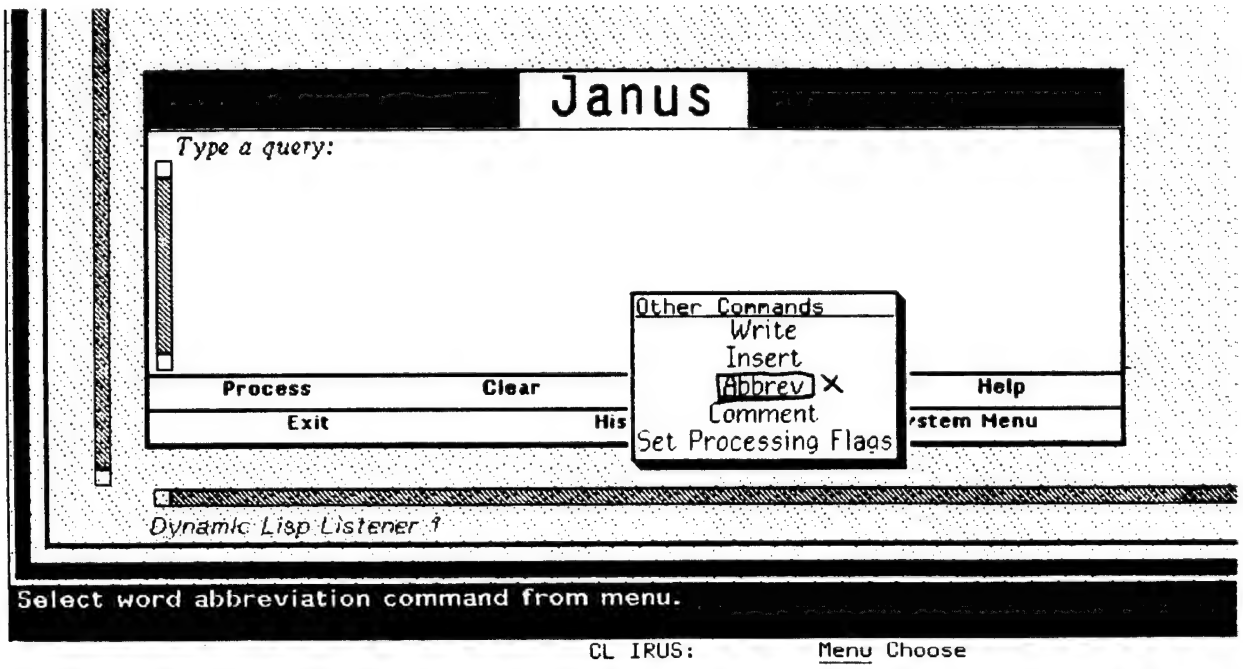


Figure 13: Select **Abbrev** command

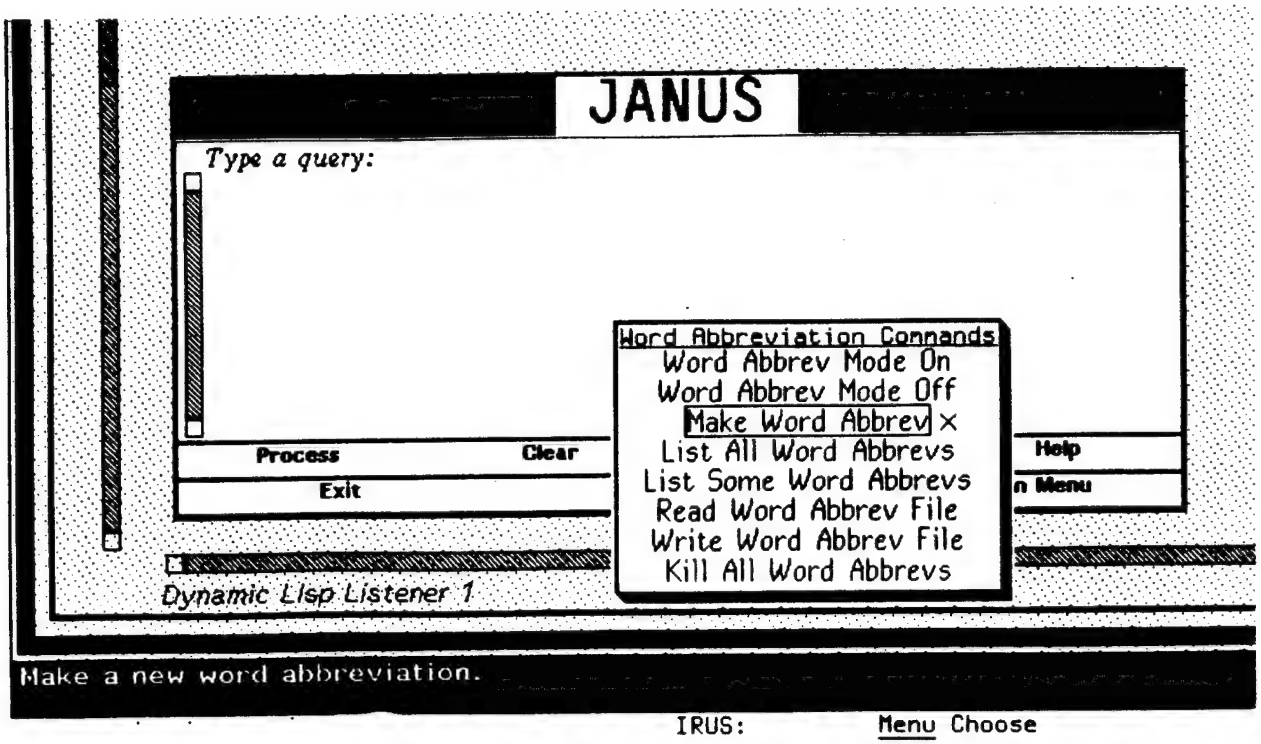


Figure 14: Select **Make Word Abbrev** command

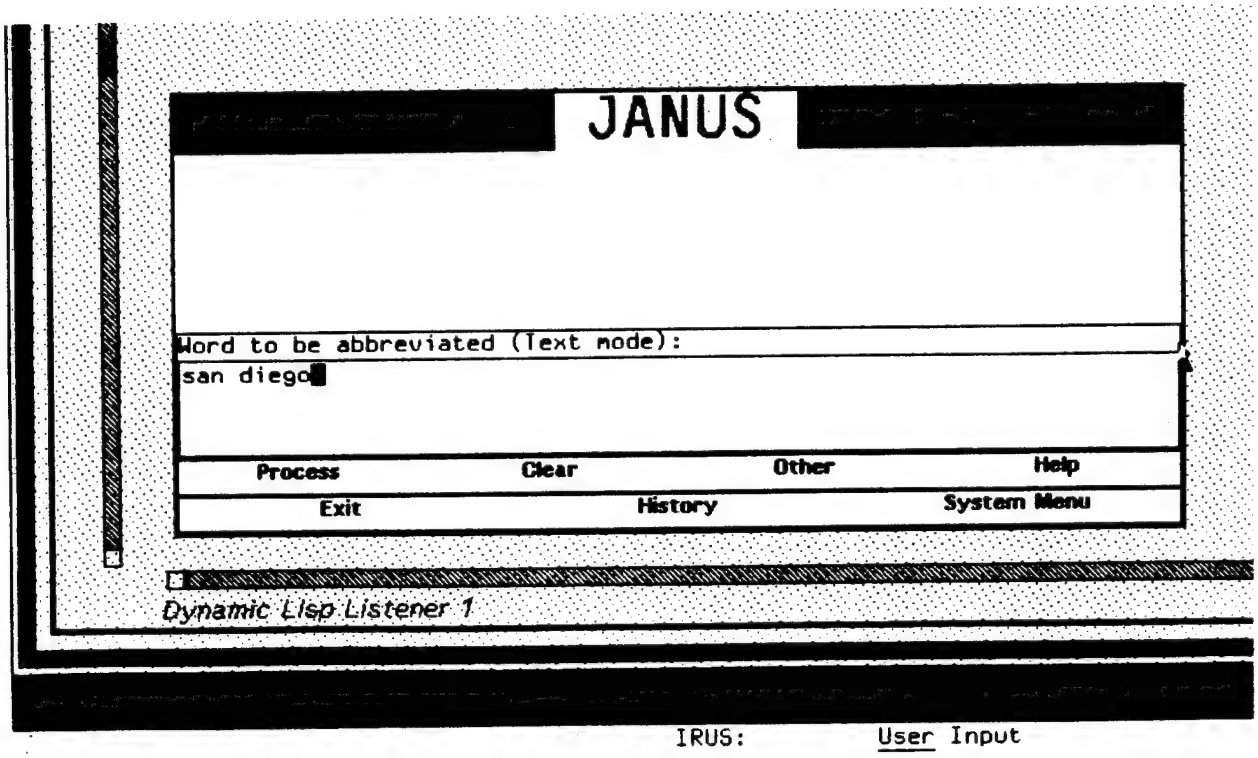


Figure 15: Prompt for word to be abbreviated

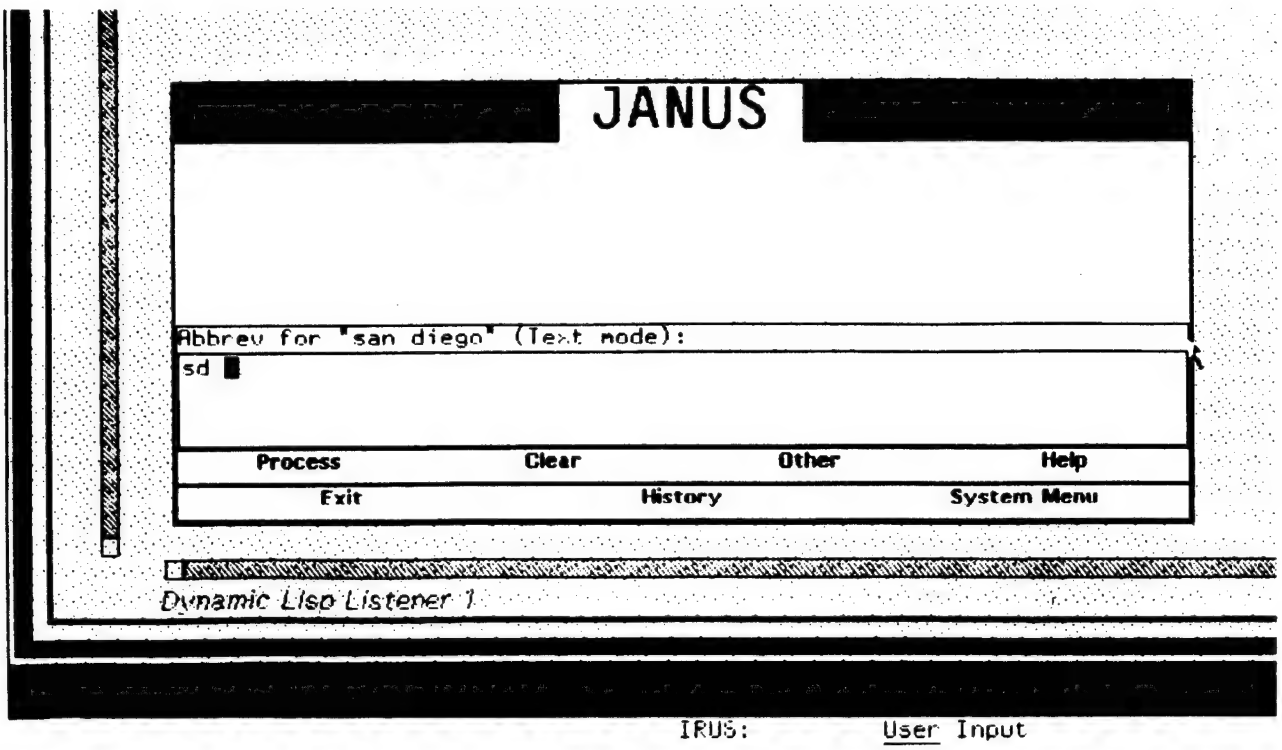


Figure 16: Prompt for abbreviation

To insert the abbreviation, the word abbreviation mode is turned on by selecting **Word Abbrev Mode On** from the abbreviation pop-up menu. Then when the abbreviation is typed and followed by a space or carriage return, the abbreviation is expanded into the full word (see Figures 17 through 19).

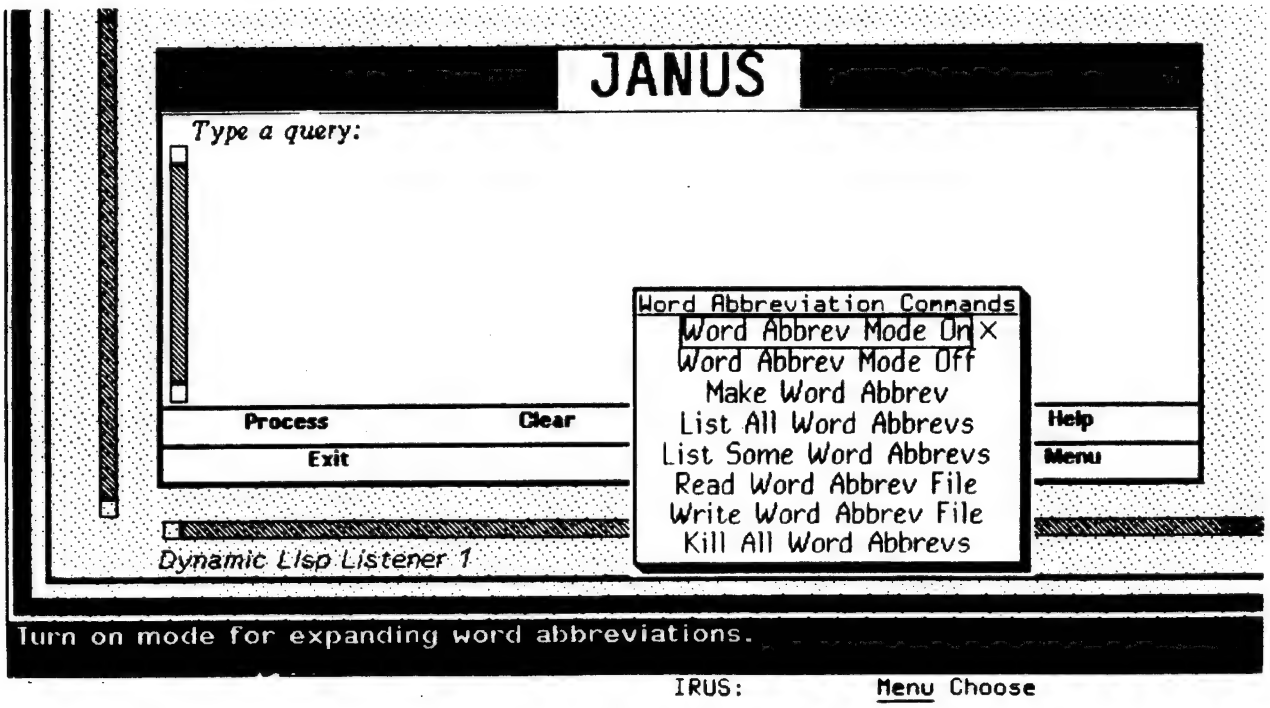


Figure 17: Turn on word abbreviation

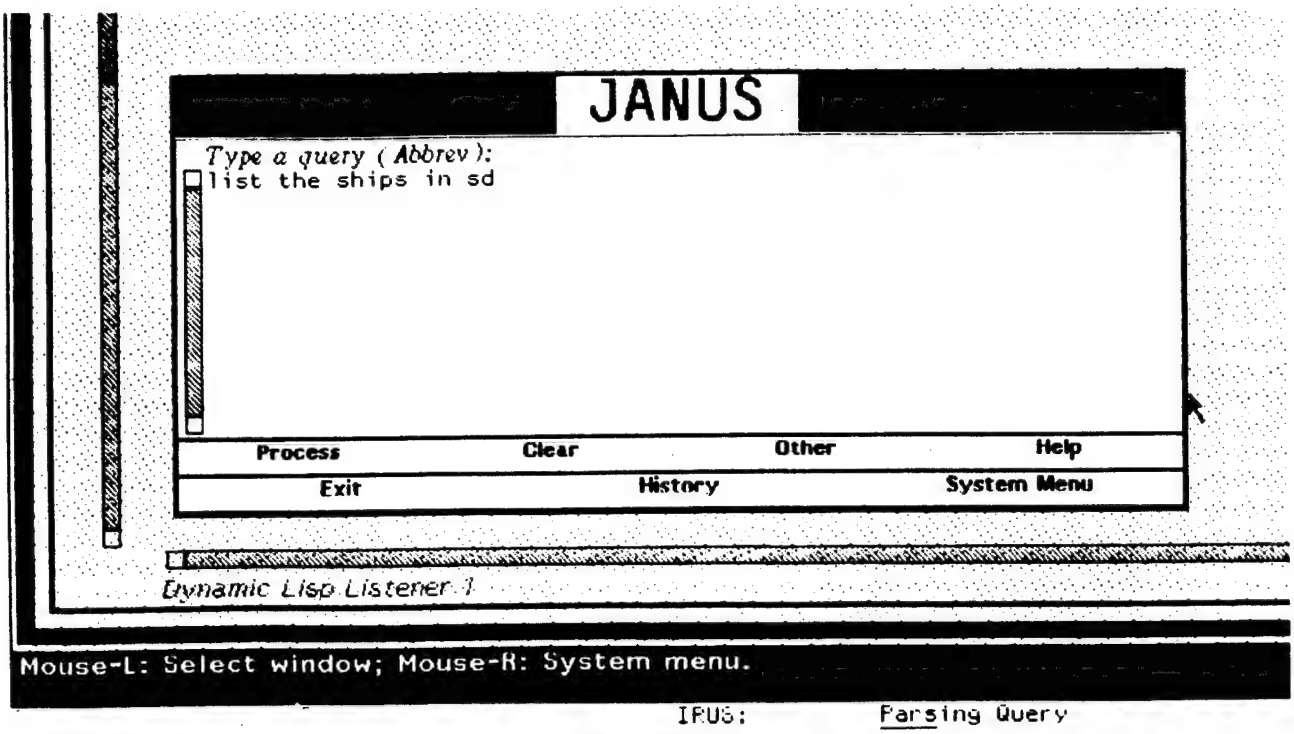


Figure 18: Enter abbreviation

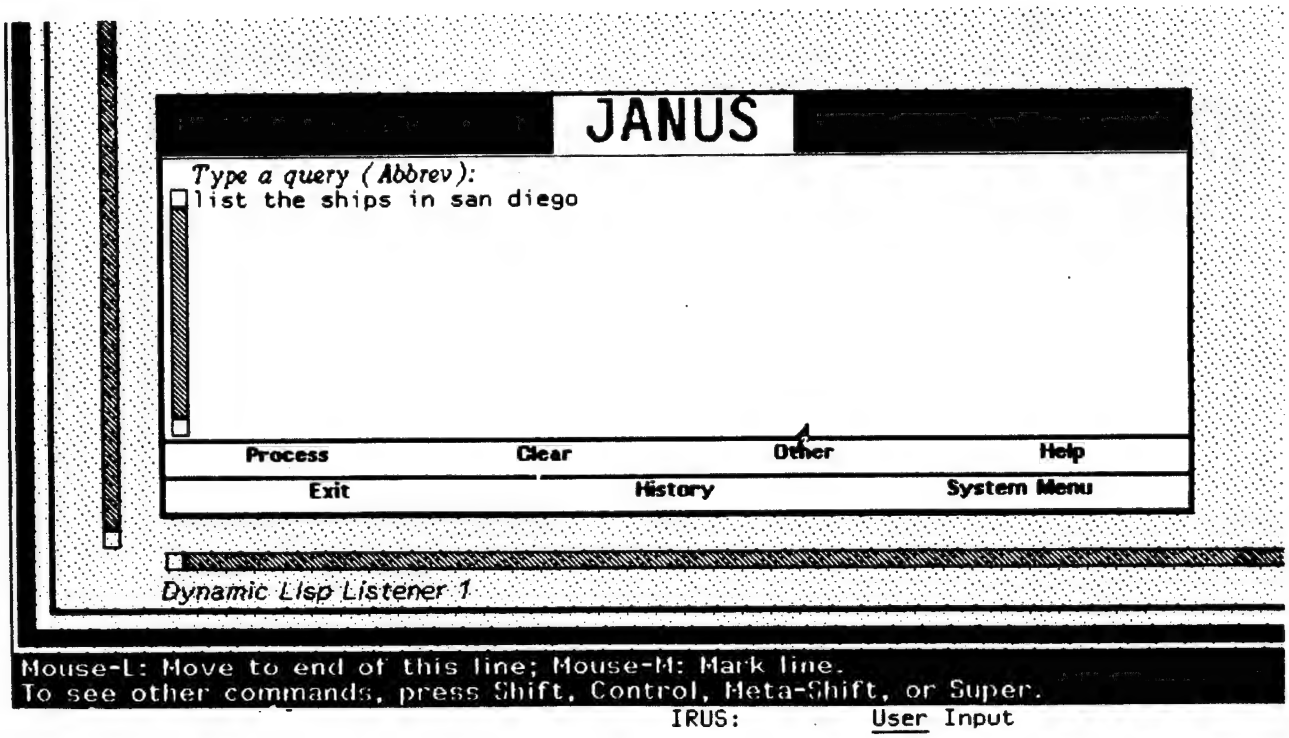


Figure 19: Word substituted for abbreviation

Should the end-user want to see a list of queries processed thus far in the session, he or she selects the **History** command, and a pop-up history window appears (see Figure 20). Queries then can be entered into the Editing pane not only by using the keyboard, but by selecting one of the queries in the pop-up History window (see Figures 21 through 23). Queries also could be entered using any combination of typing, reading a file, or selecting a graphic object with the mouse. The query, "List the ships in San Diego", could have been composed by reading a help file or menu of possible queries and selecting "list", typing "the ships", and selecting the visual object corresponding to the port "San Diego" on a chart display, provided that the graphic objects or pieces of text are mouse selectable and return a string.

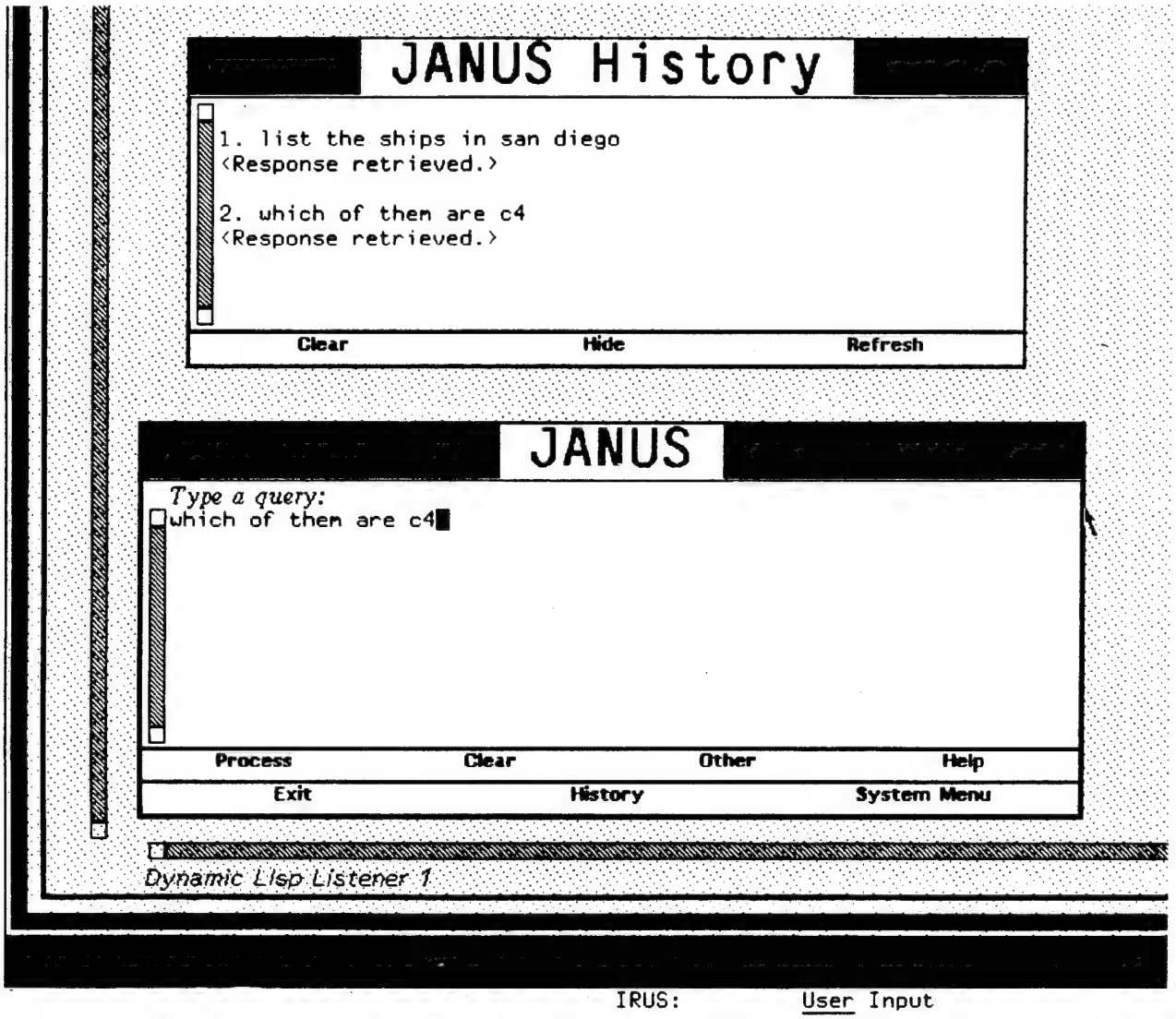


Figure 20: Pop-up History window

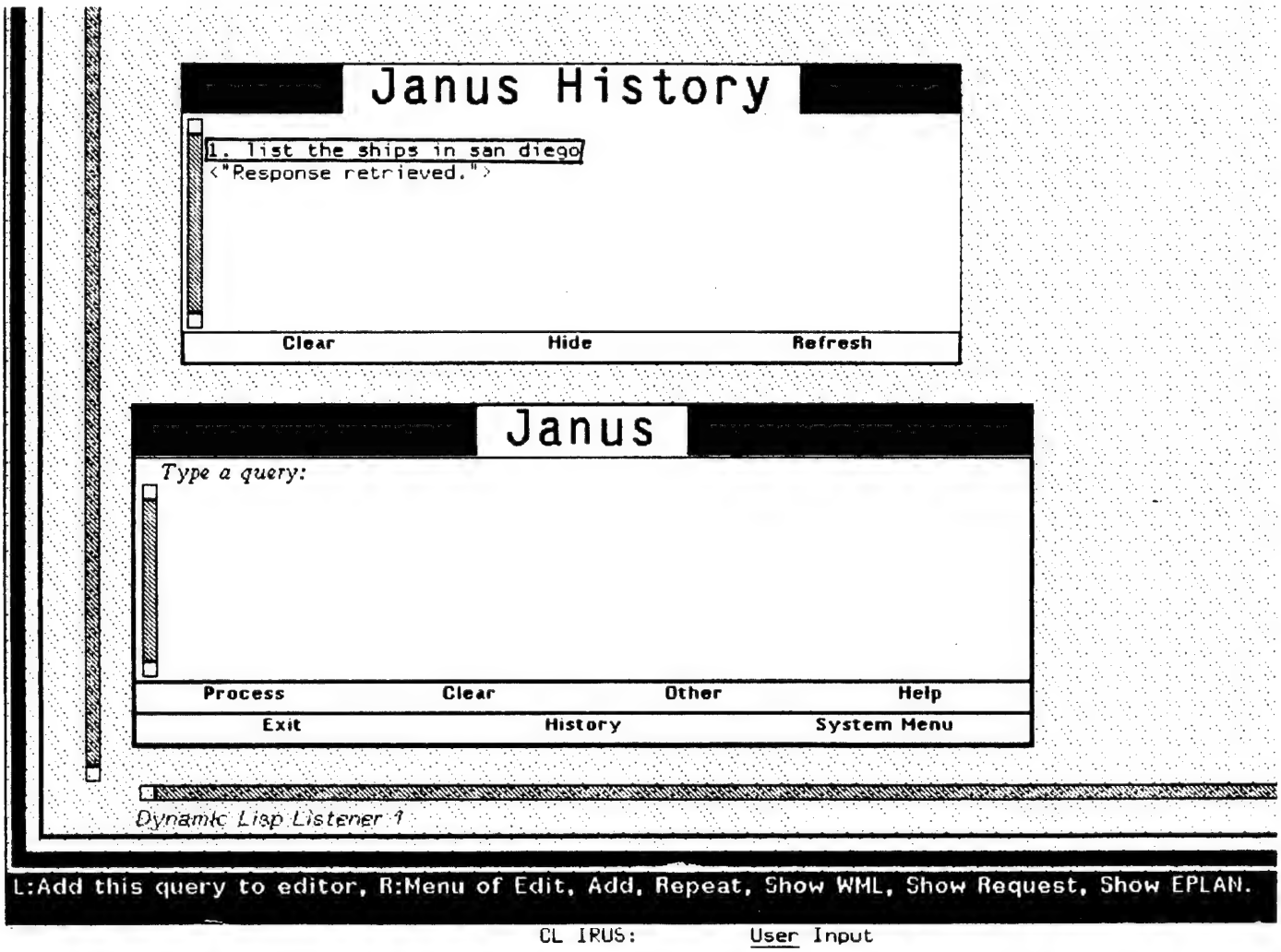


Figure 21: Select query from History

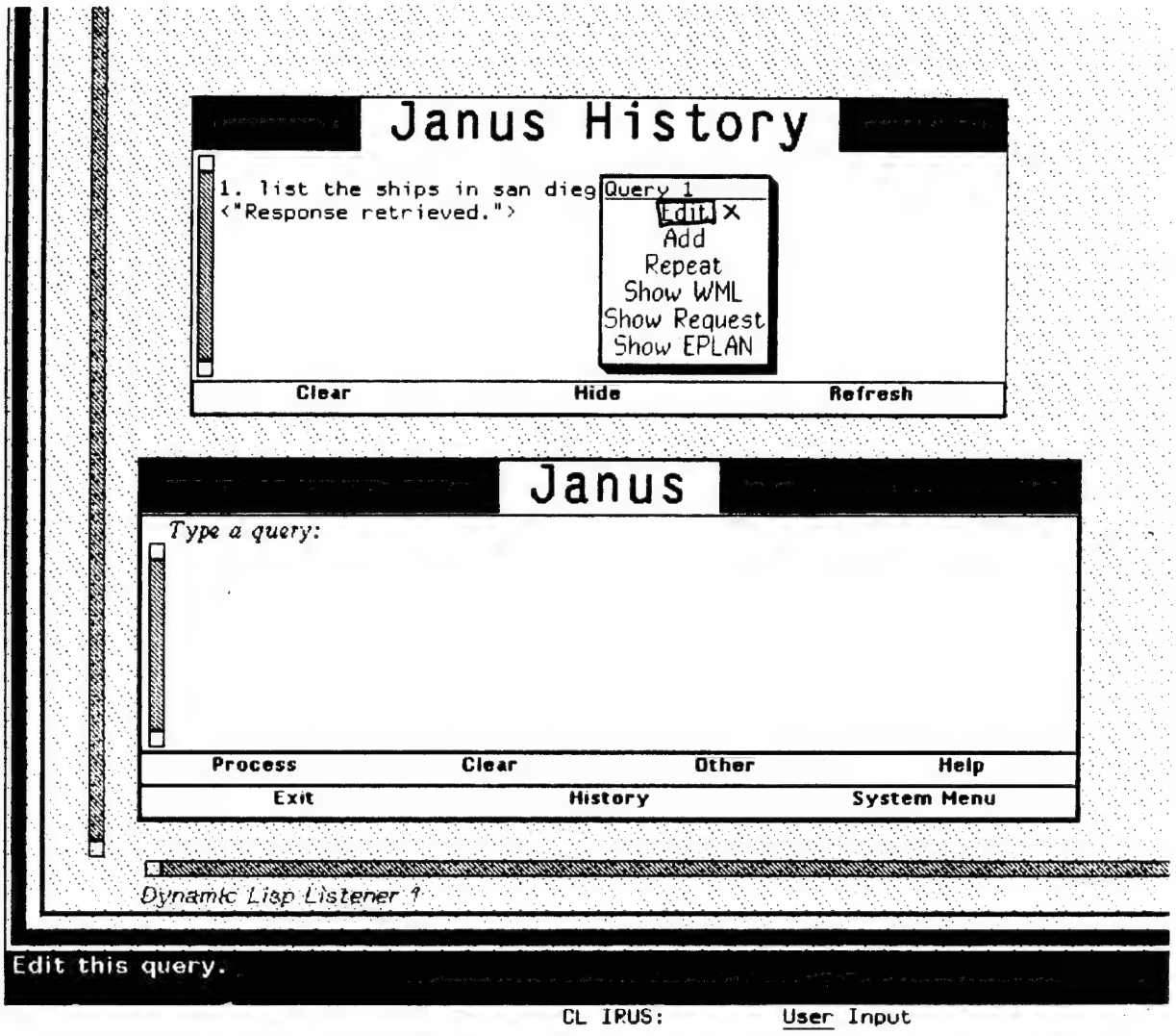


Figure 22: Select **Edit** command for query

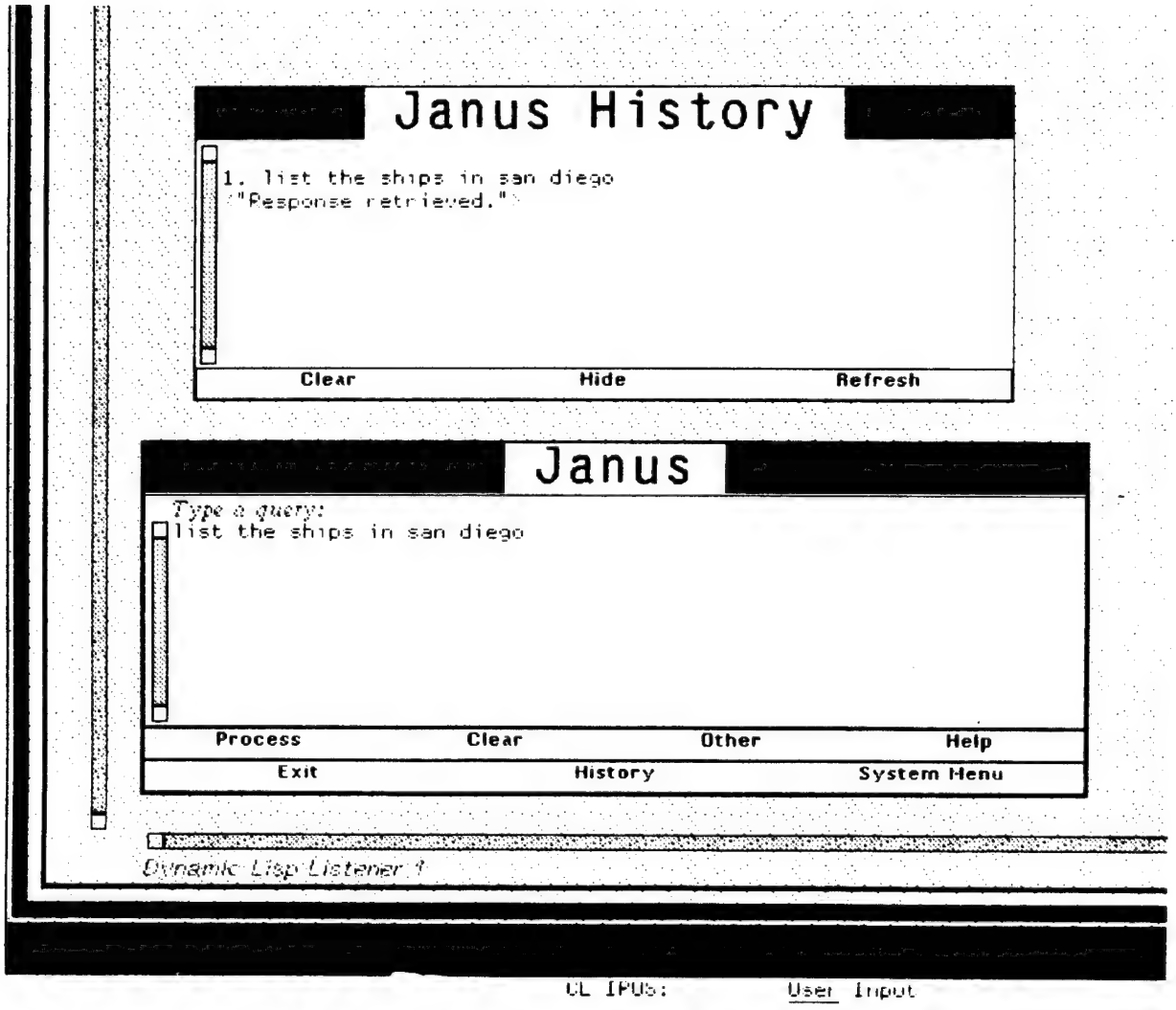


Figure 23: Query added to Editor pane

4.2 Developer Interaction

All the functionality supplied to the end-user is available to the developer. In addition, capabilities have been provided for system development, debugging, testing, and updating IRUS. The following scenario illustrates the use of these capabilities.

The full screen developer configuration is selected, and a query is entered into the Editing pane (see Figure 24). The developer is often interested in stopping the processing at an intermediate stage and may select a stage on the processing stages menu located just below the Editing pane. In this example, the WML stage has been selected. The query is processed, and the results are shown in the Interaction pane and in an abbreviated form in the History pane (see Figure 25). The result, WML in this example, can be edited by clicking on the word "WML" in the Interaction pane, then clicking on **Edit this object** in the menu that pops up. The WML Lisp expression is added automatically to the Editing pane, and the mode is set to WML. (See Figures 26 through 28.) Should the developer then want to proceed with processing to the next step, Request can be selected on the processing stages menu, and the query processed. The resulting Request is displayed in the Interaction pane and in abbreviated form in the History pane (see Figure 29).

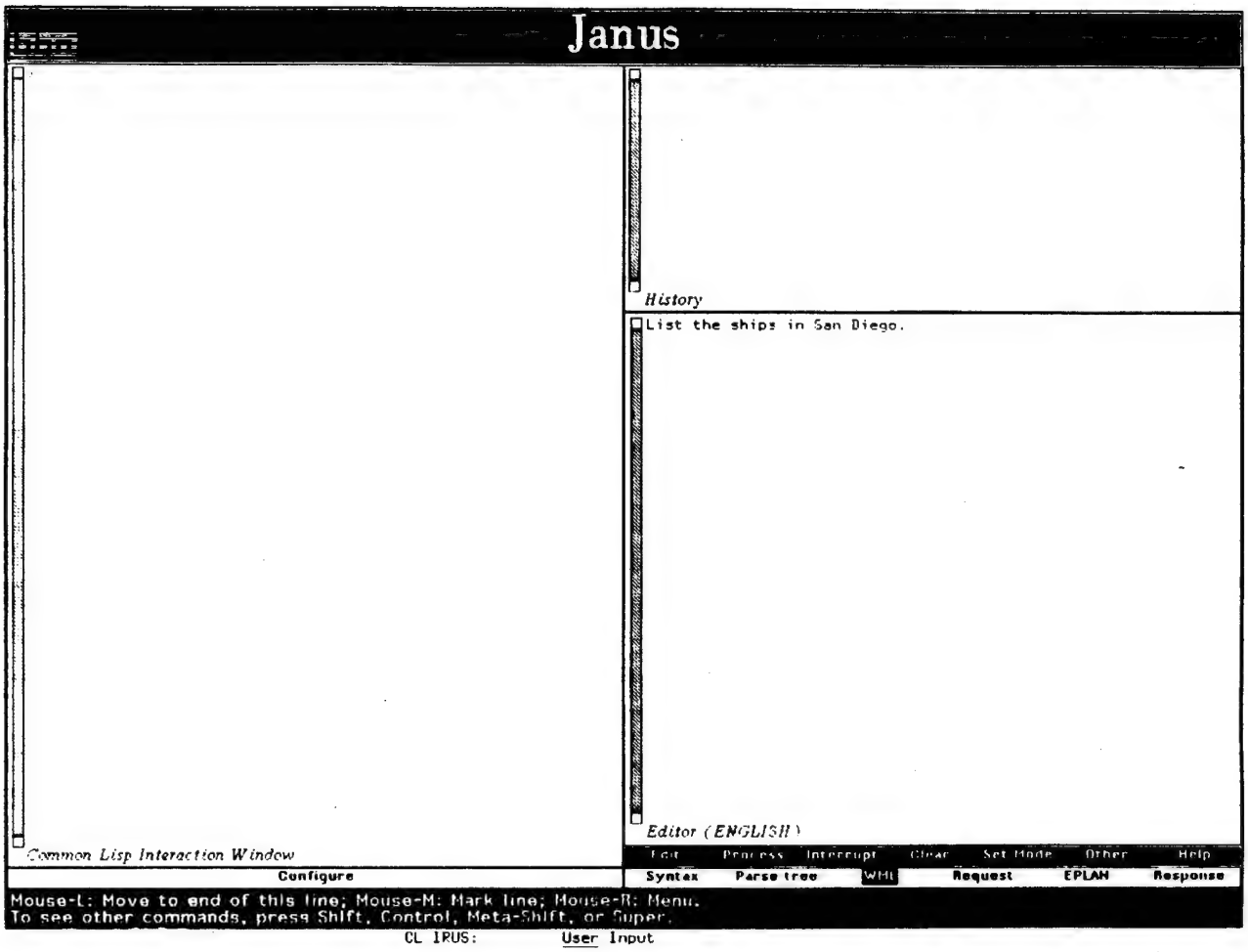


Figure 24: Process query in developer configuration

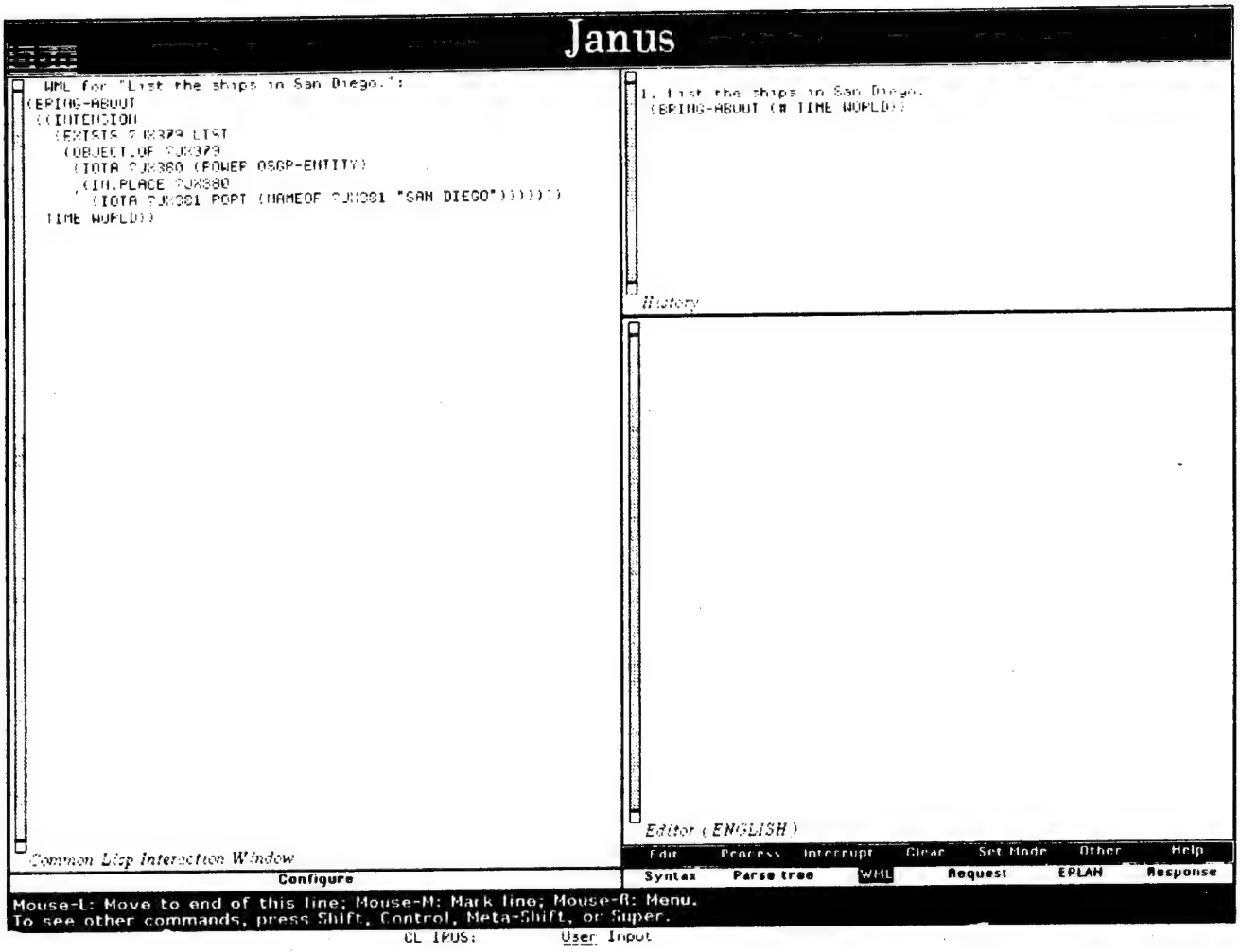


Figure 25: Query result displayed in Interaction and History panes

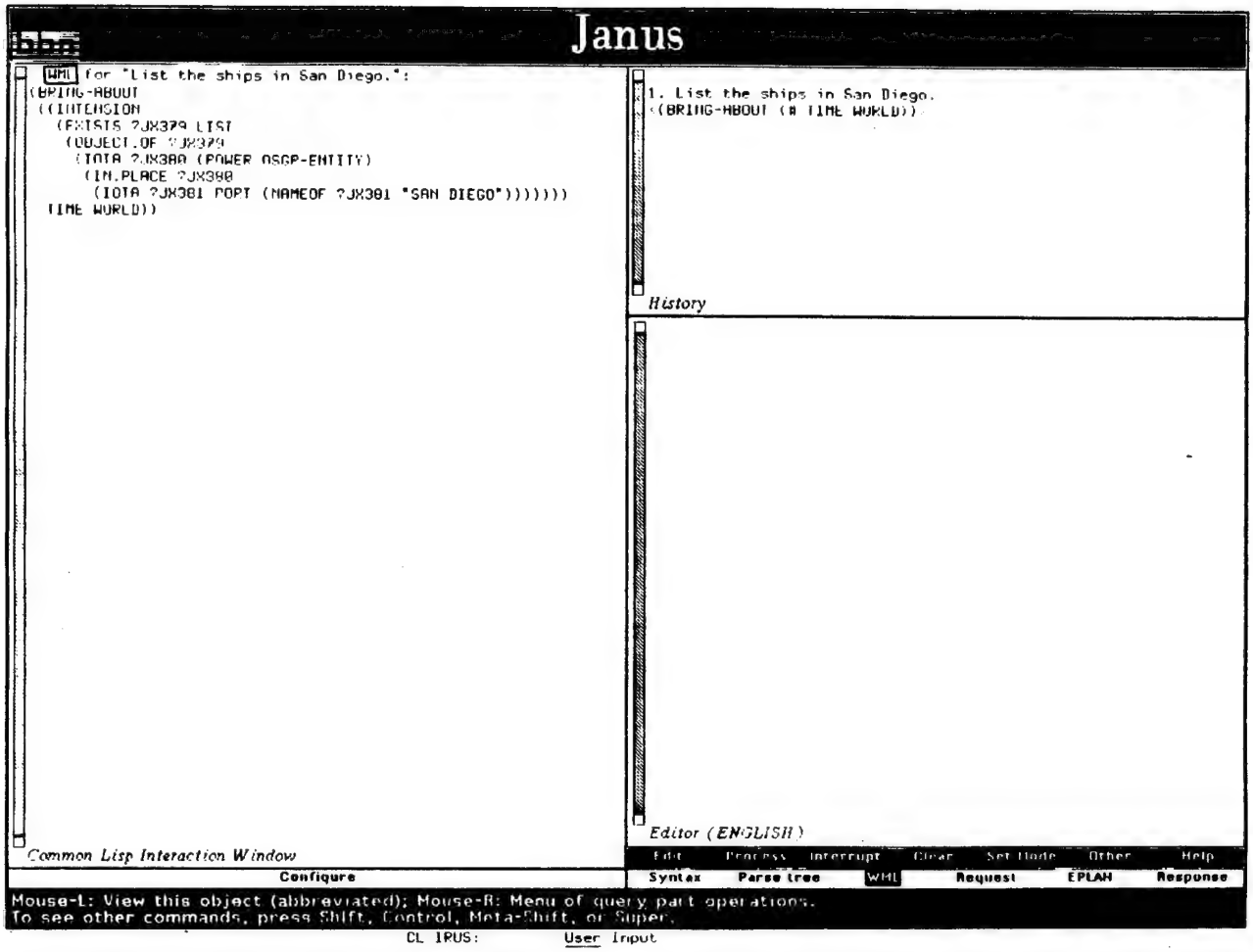


Figure 26: Select WML from Interaction pane

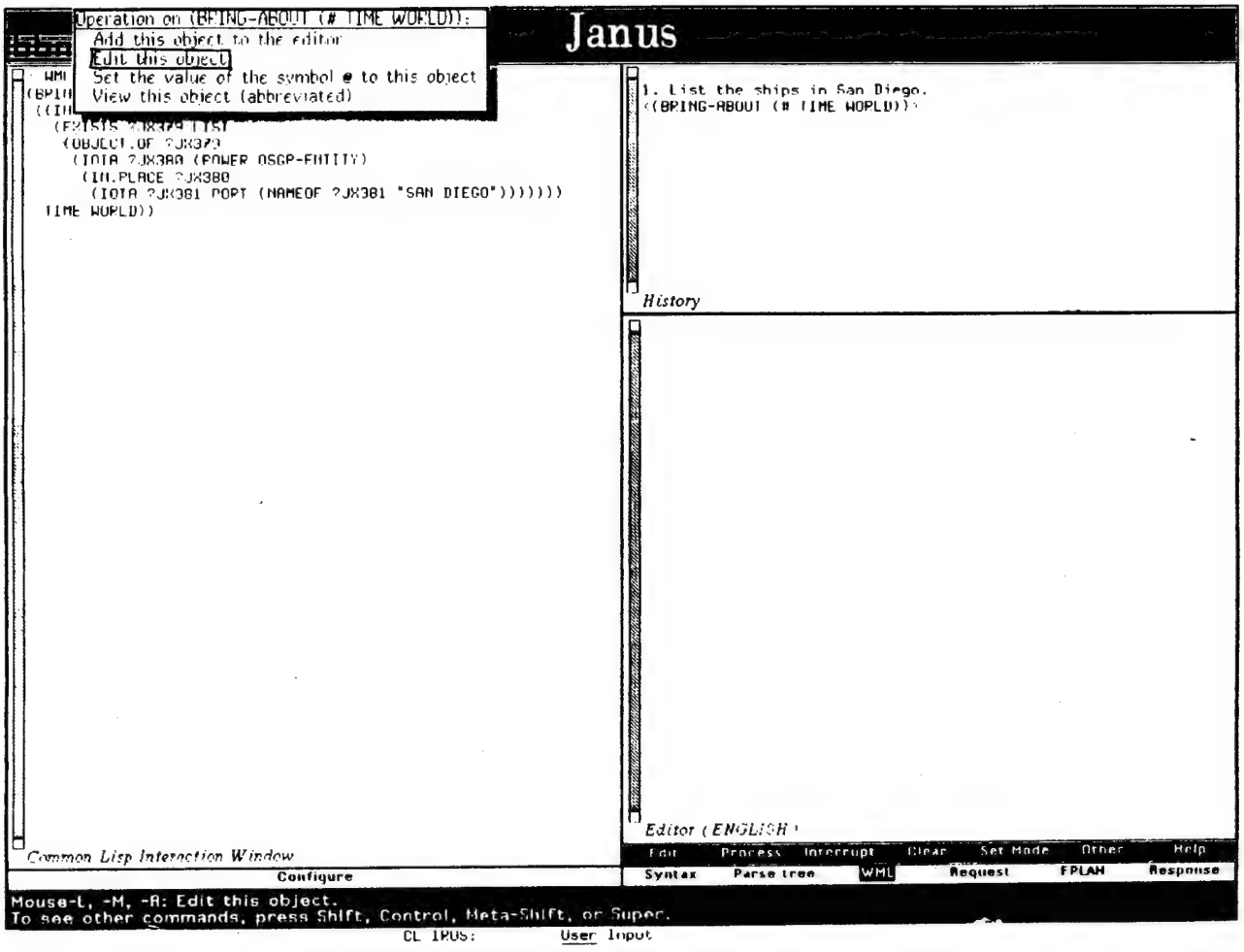


Figure 27: Select Edit this object command for WML

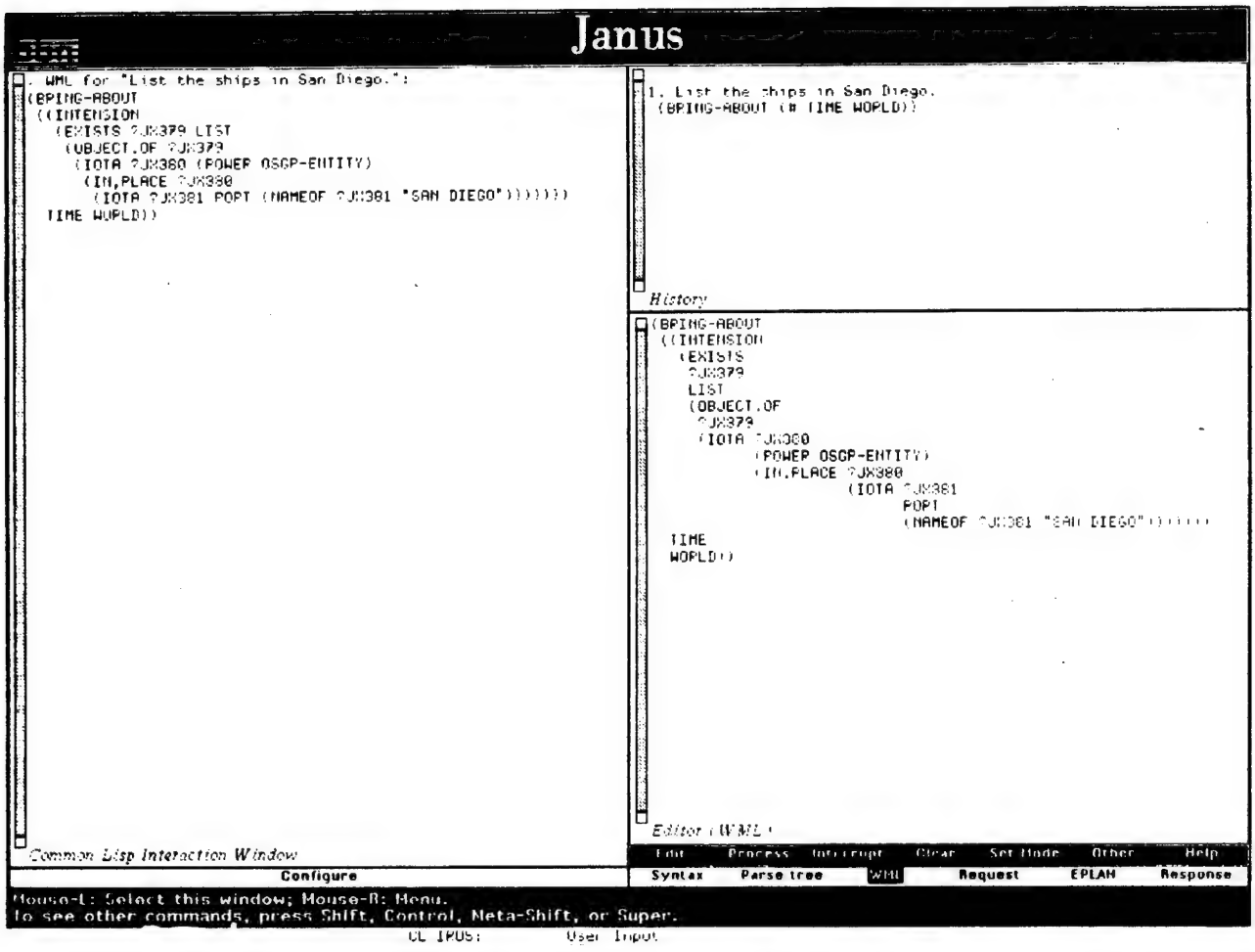


Figure 28: WML added to Editor pane

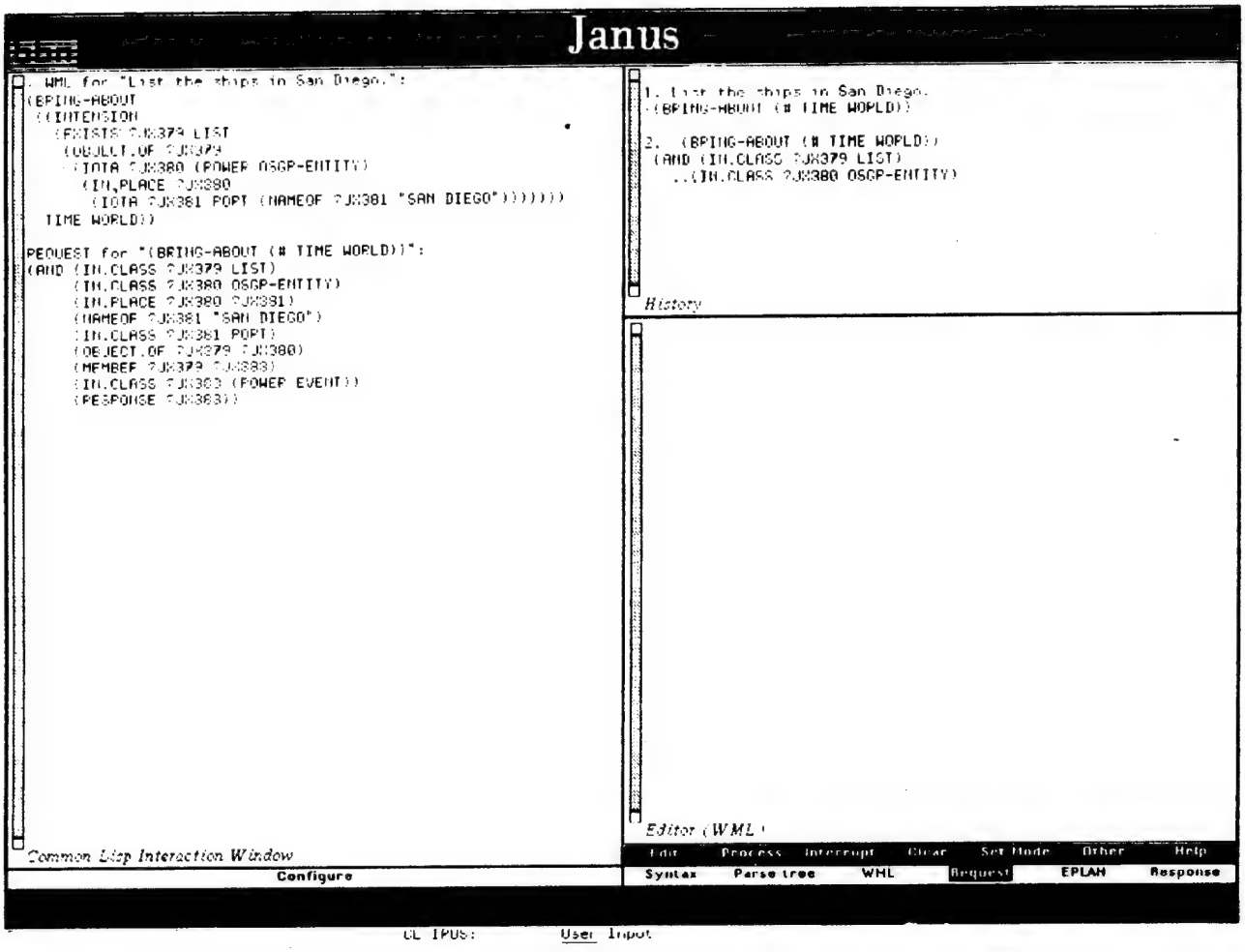


Figure 29: Request displayed in Interaction and History panes

The developer can display all processed stages of a query by clicking on the query in the History pane (see Figures 30 through 32). The query stages, displayed in the Interaction pane, are mouse-sensitive and can be operated upon (e.g. edited or viewed in more detail) by selecting an operation from a pop-up menu that appears when the stage is selected with the mouse (see Figures 33 through 35).

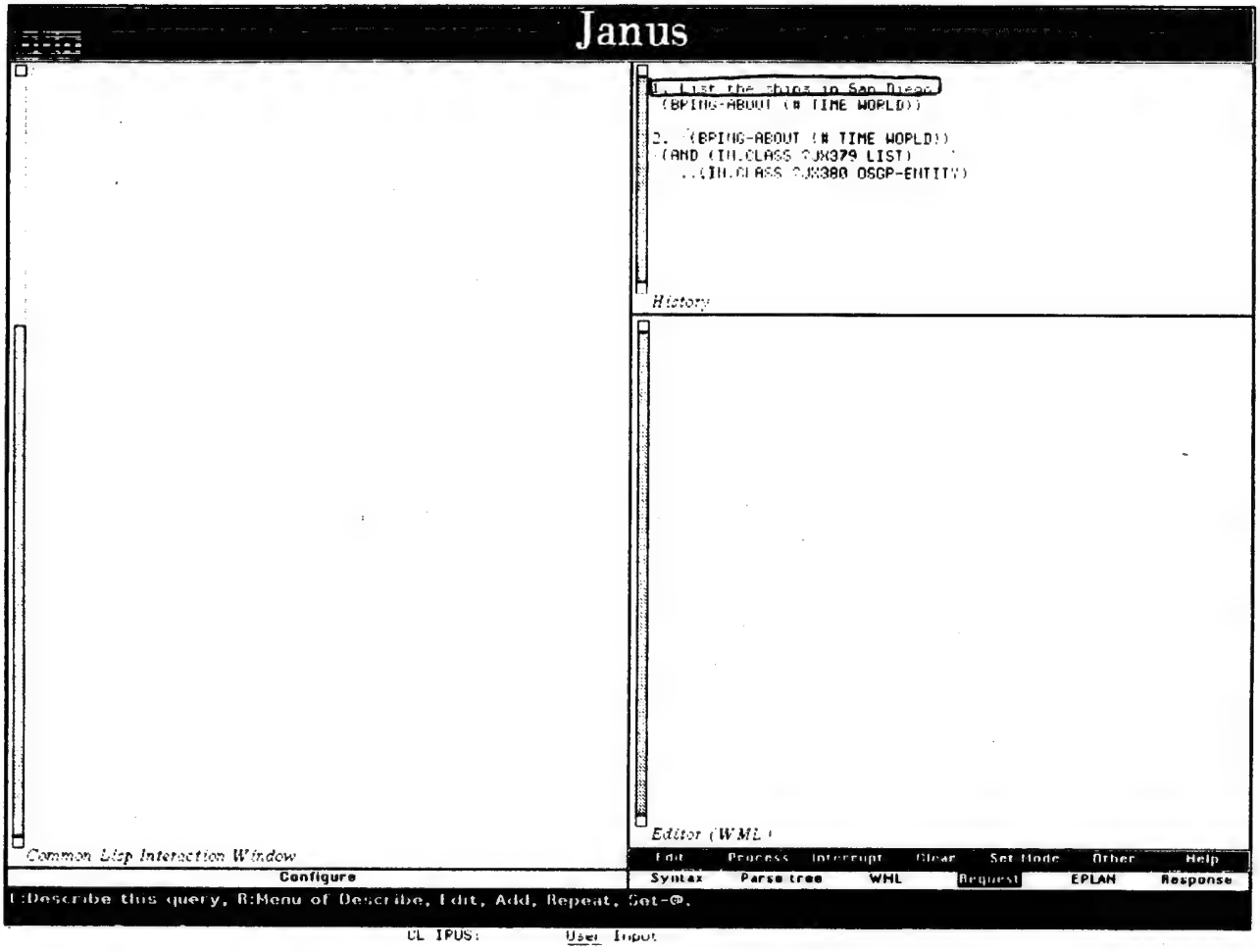


Figure 30: Select query from History pane

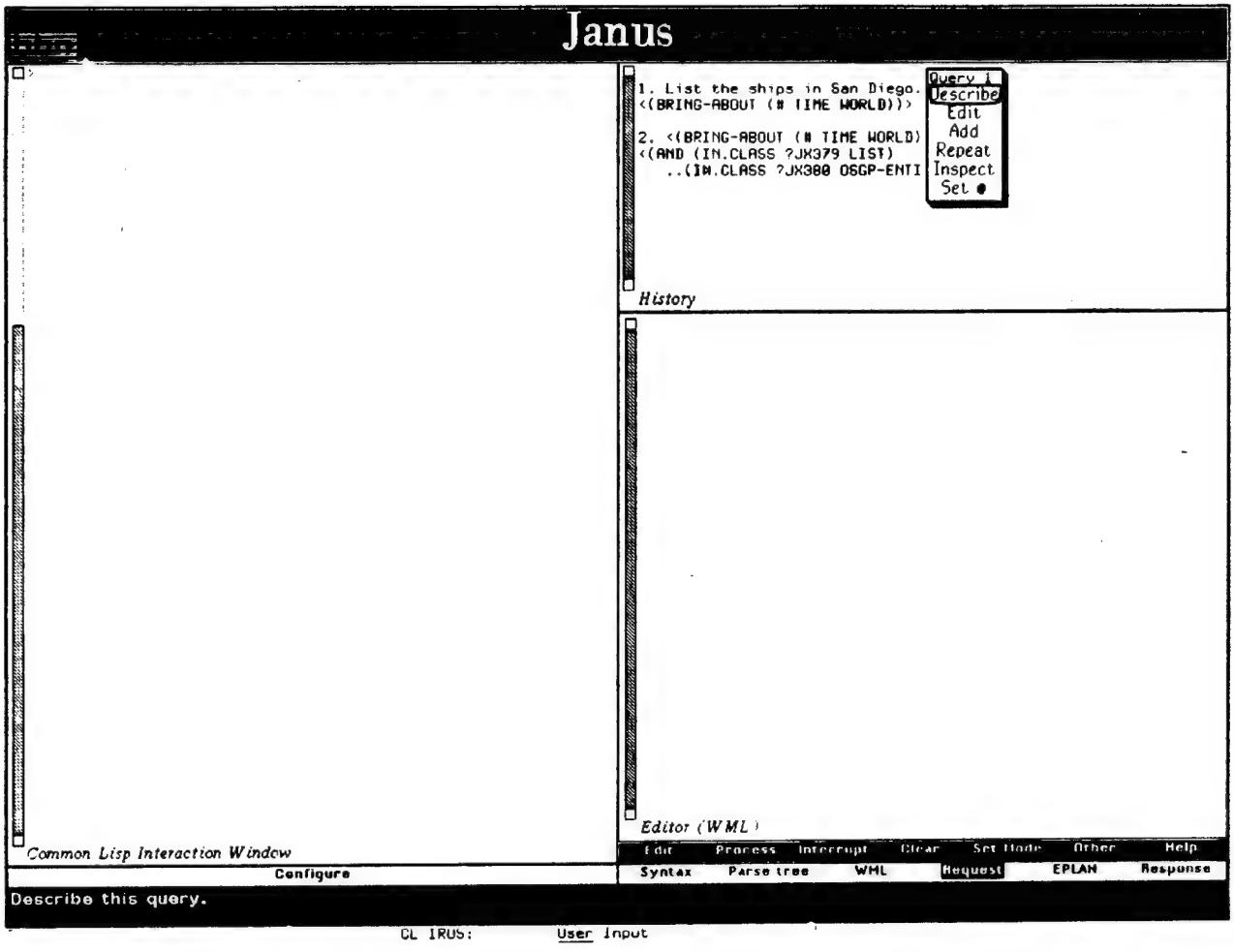


Figure 31: Select **Describe** command for query

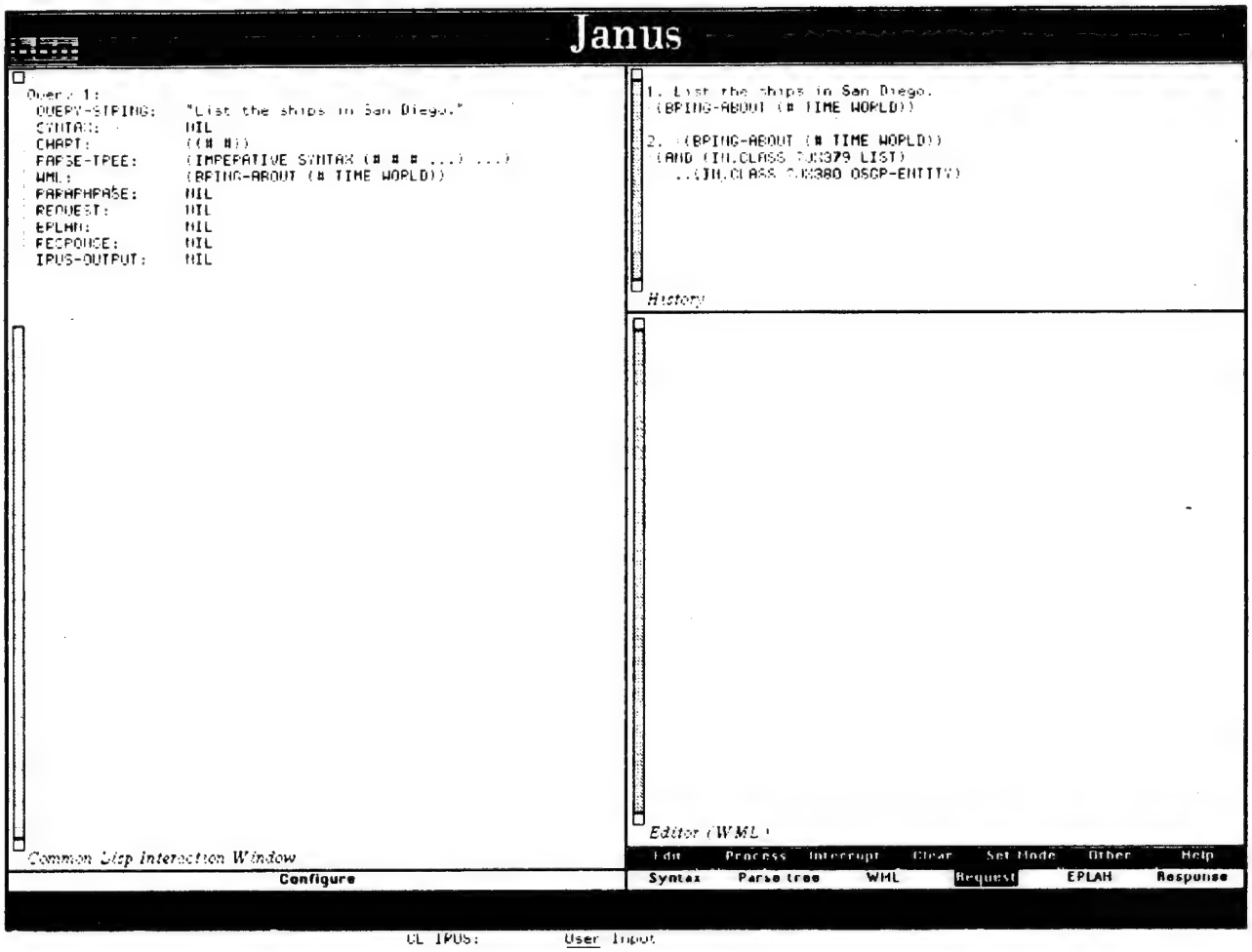


Figure 32: Query object described in Interaction pane

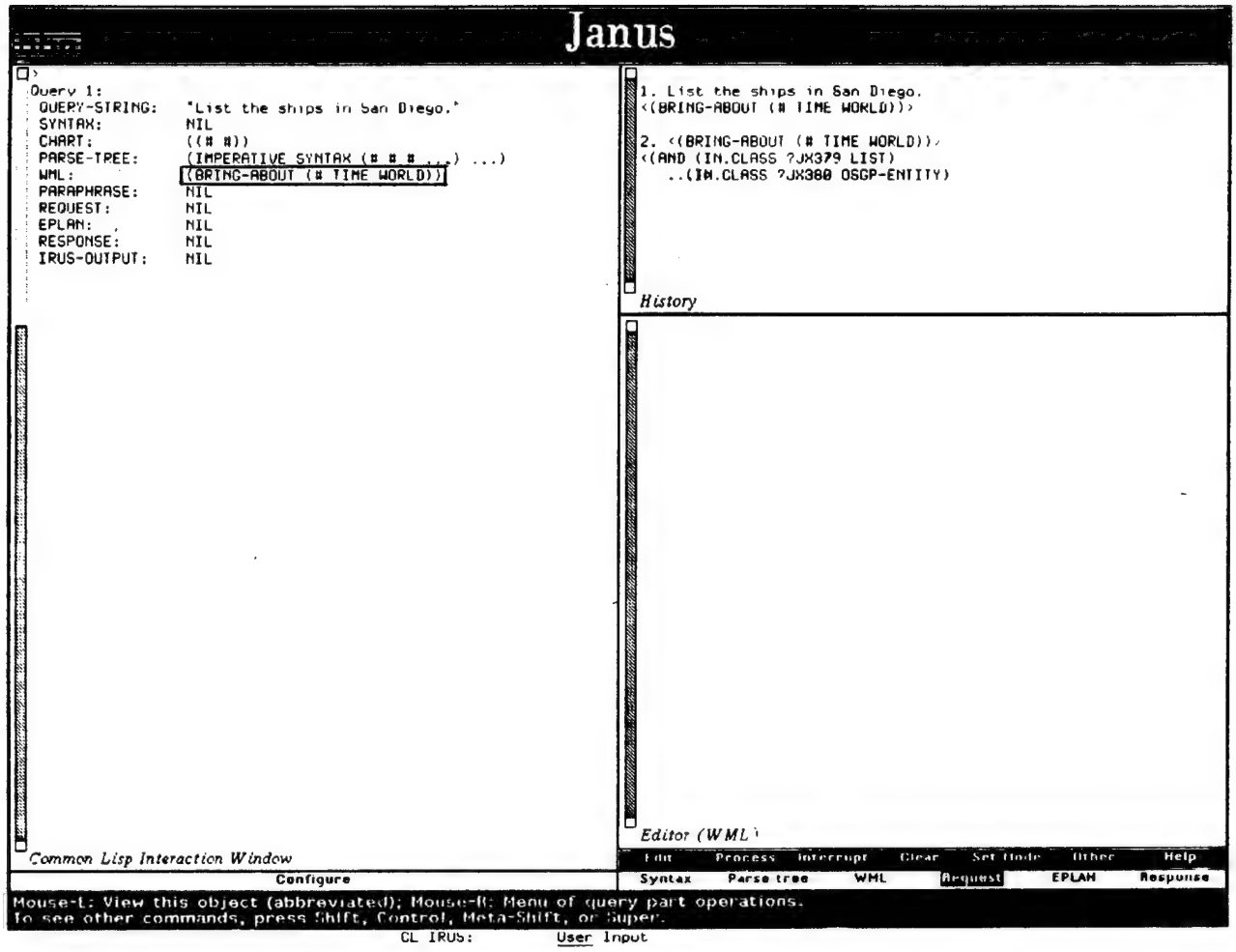


Figure 33: Select WML from Interaction pane

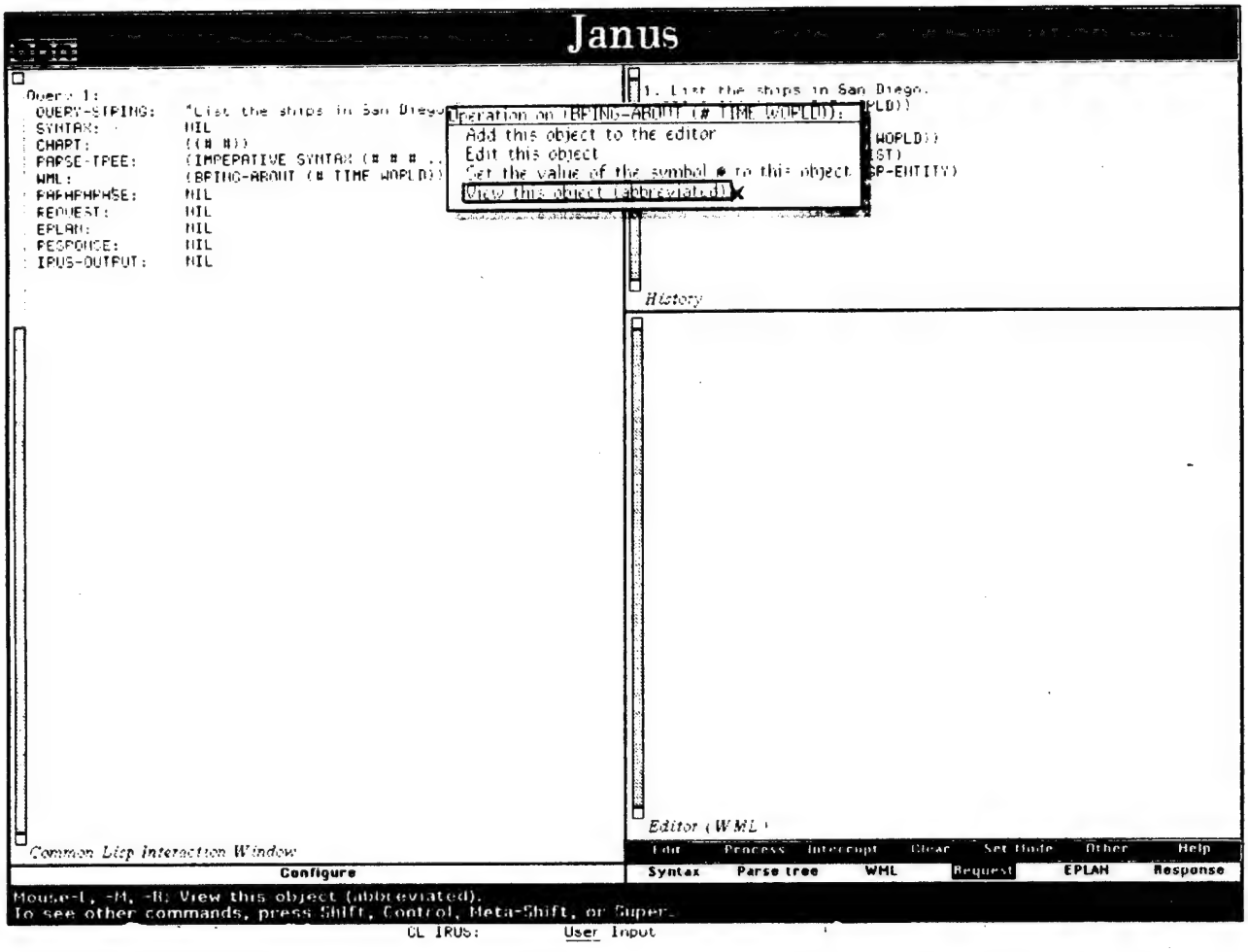


Figure 34: Select View this object command for WML

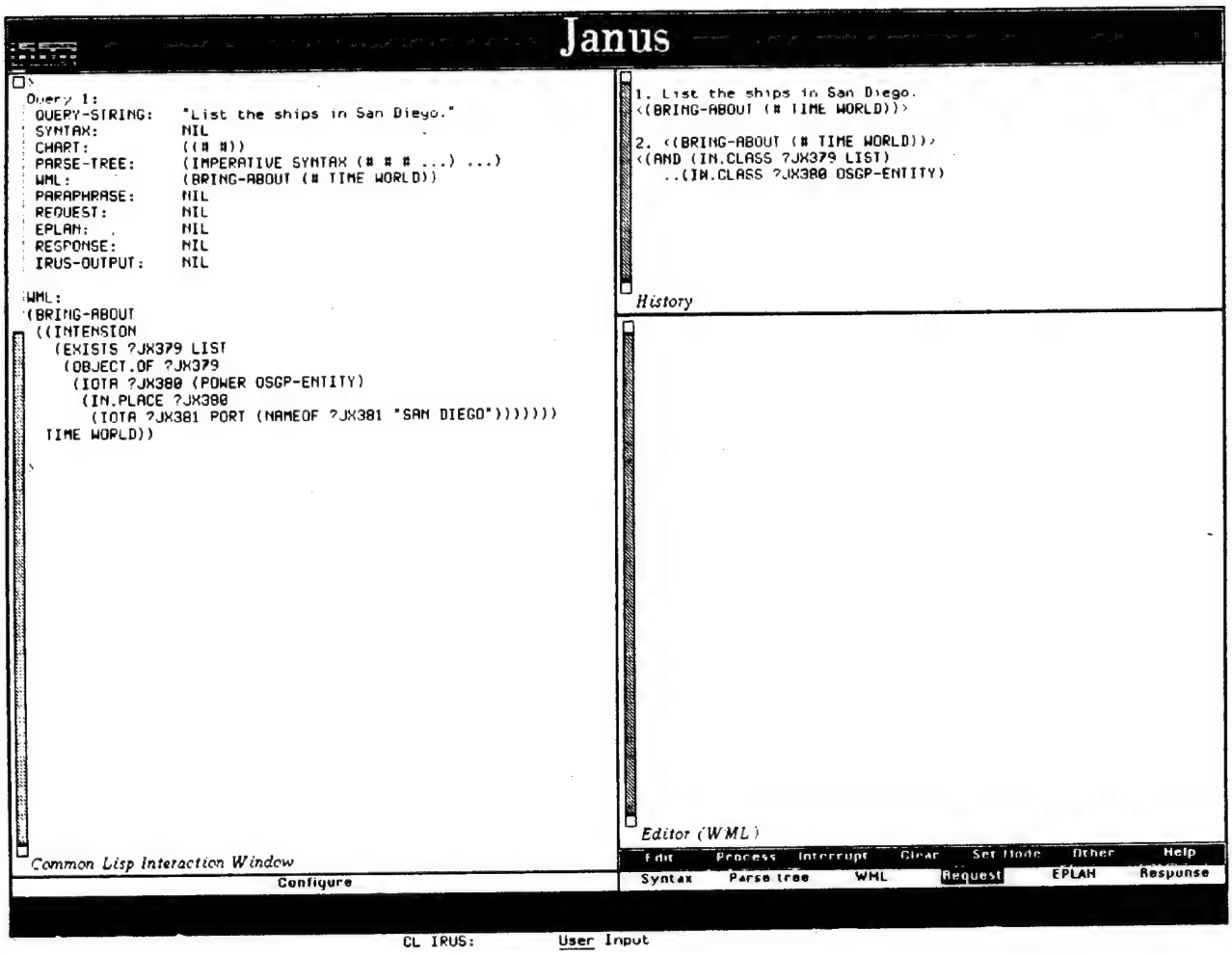


Figure 35: WML displayed in Interaction pane

4.3 Screen Management

The IRUS interface provides both developer and end-user with a variety of configurations and screen manipulation tools. In this section, we first describe two other developer interfaces, and then show how the pop-up windows can be modified to fit in with other interfaces.

In addition to the developer interface described in the previous section, the developer is provided with configurations for editing the rules used for interpreting English sentences (the IRACQ configuration) and editing IRUS' dictionary of words (the IRUS Dictionary configuration). These configurations are selected using the **Configuration** command (see Figure 36).

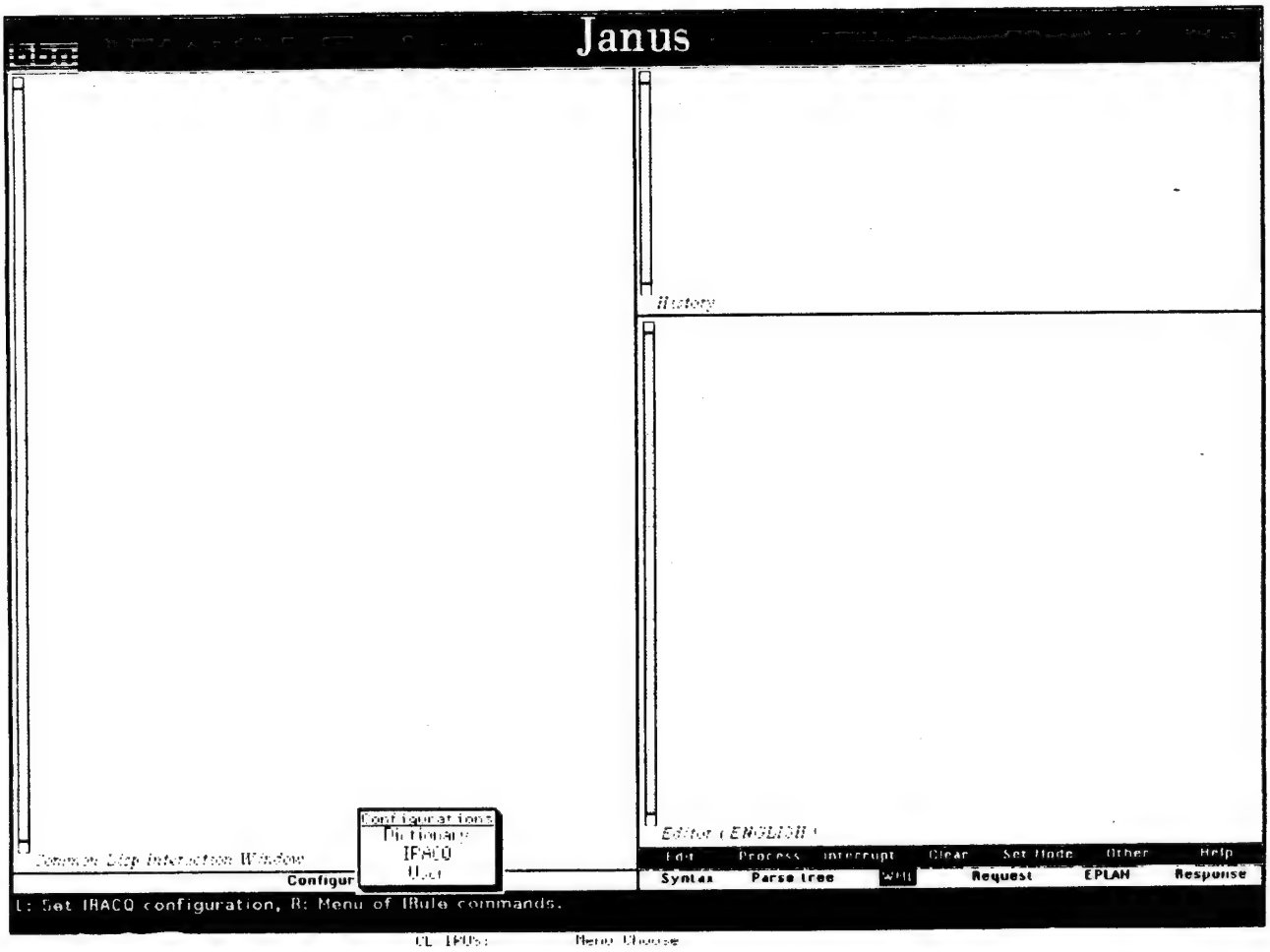


Figure 36: Configuration menu

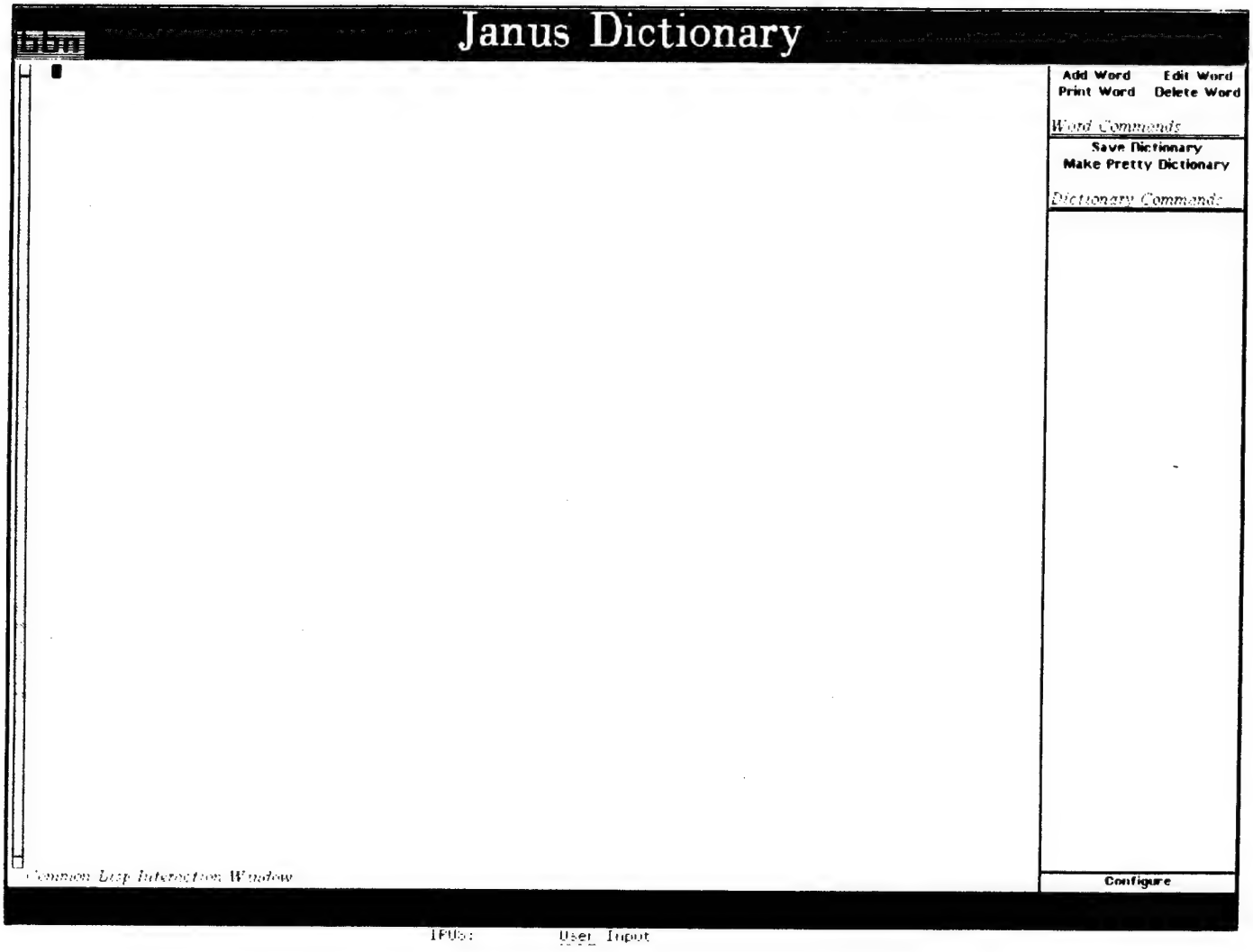


Figure 37: Dictionary Configuration

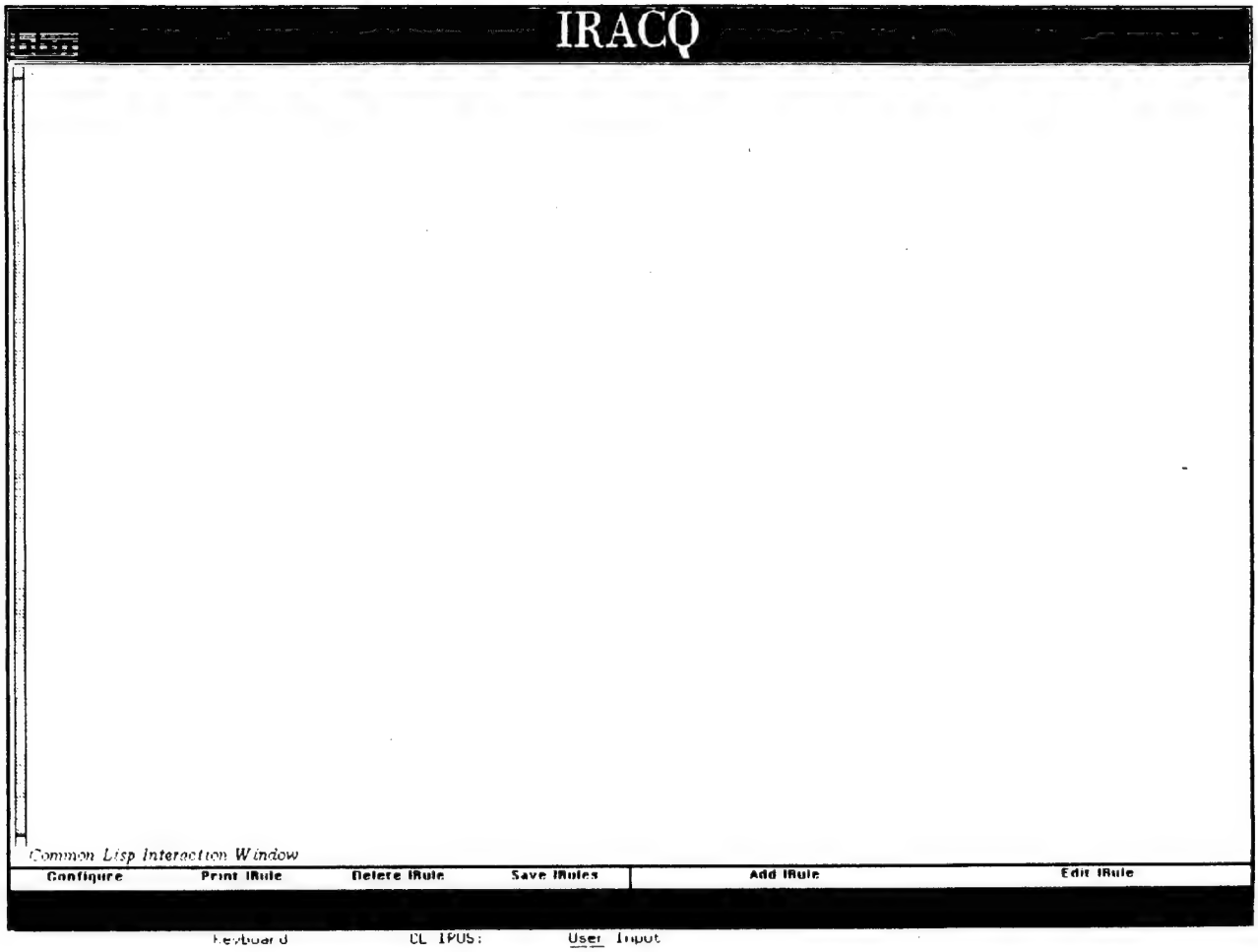


Figure 38: IRACQ Configuration

Figures 39 through 41 show how the IRACQ configuration can be used to edit IRULES. For a more complete description of IRACQ, see Ayuso, Shaked, and Weischedel (1987).

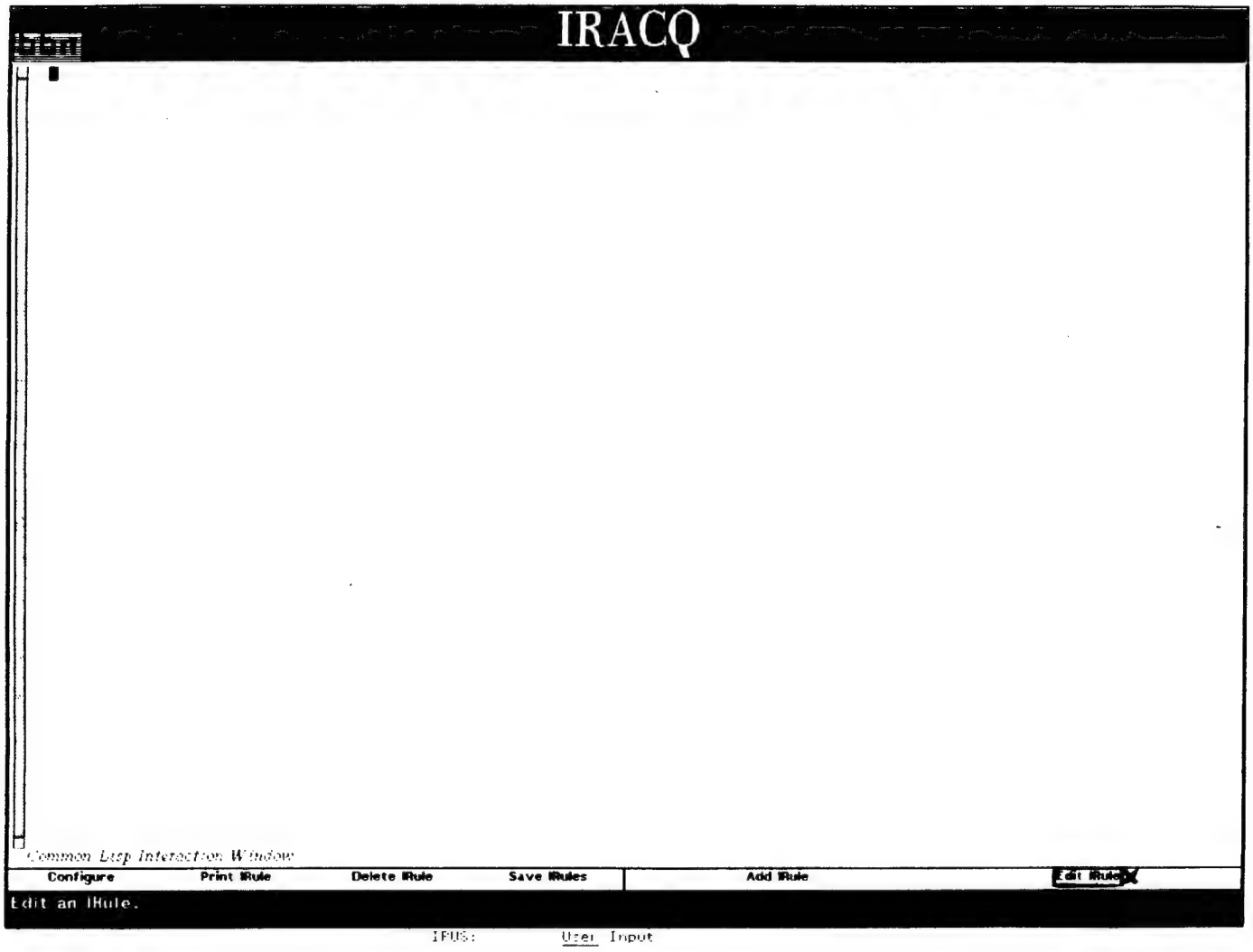


Figure 39: Select Edit Irule command

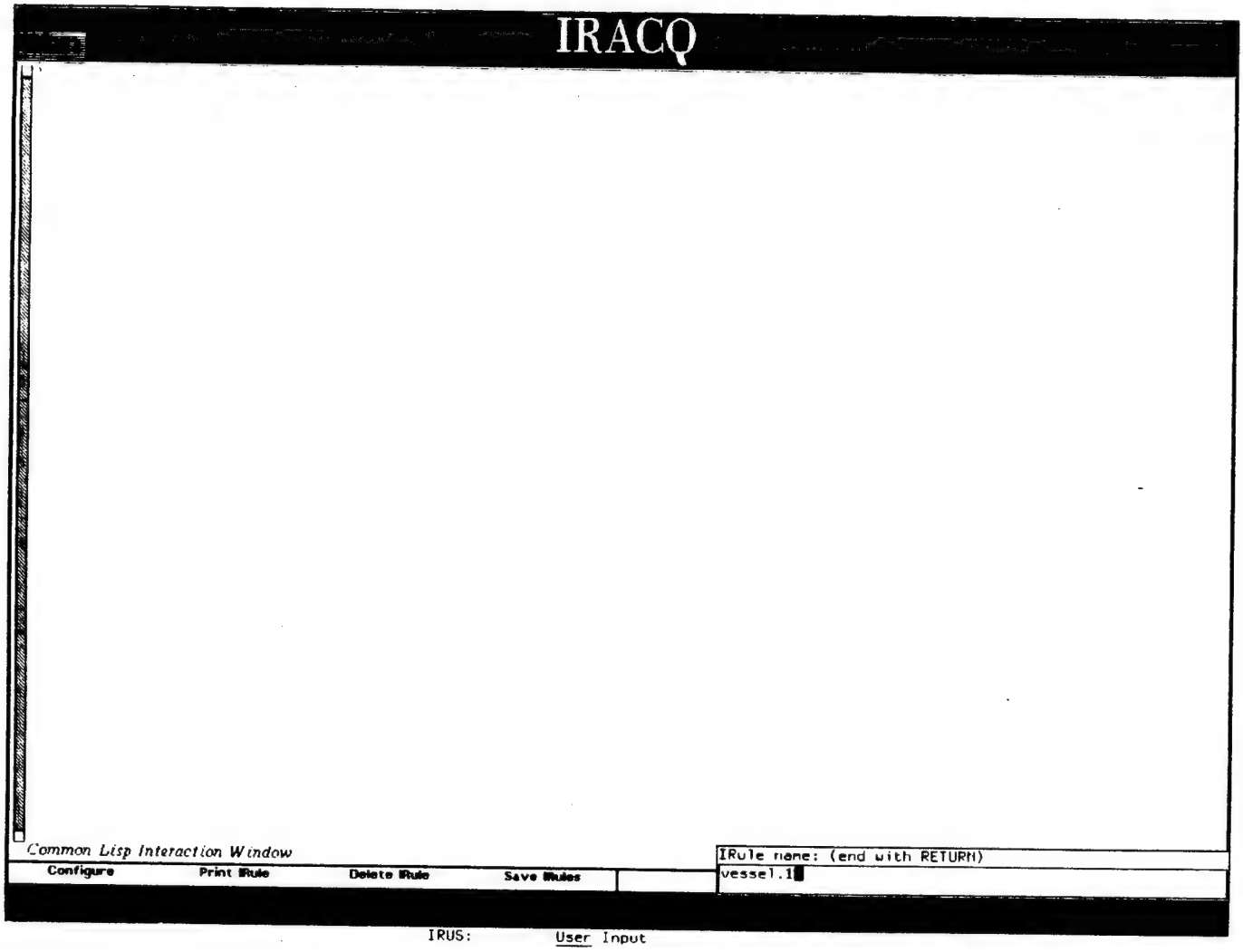


Figure 40: Enter IRule name

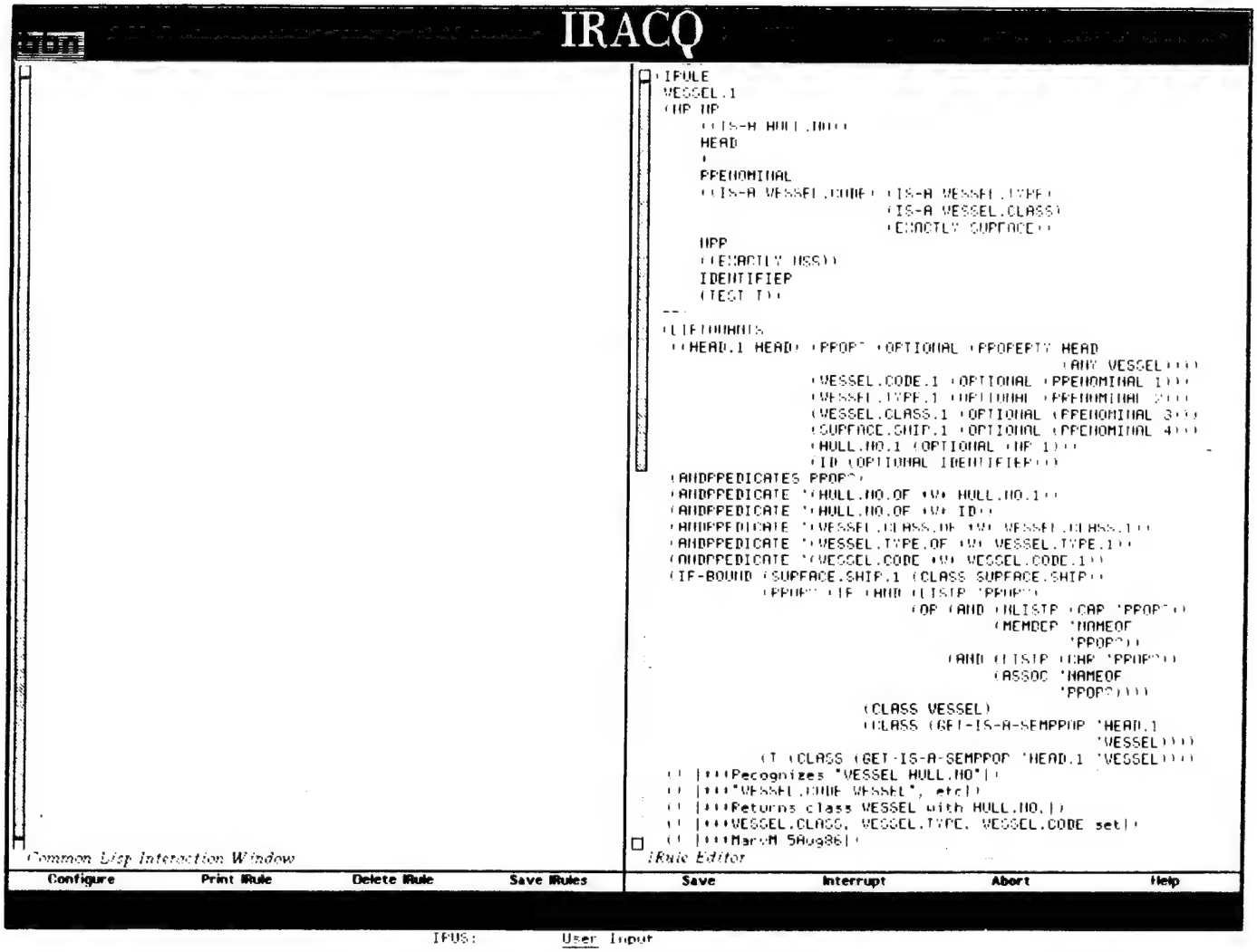


Figure 41: IRule appears in editing pane

The pop-up Interaction and History windows can be moved and reshaped dynamically to suit the user's needs. First, move the mouse to the black box at the top of the pop up window (see figure 42). As in other window operations, the command line at the bottom of the screen will give the next available operations. There are two ways to manipulate the windows: 1) directly with the mouse (clicking left to move and middle to reshape); and 2) using a menu for a more complete set of operations (clicking right) (see figure 43).

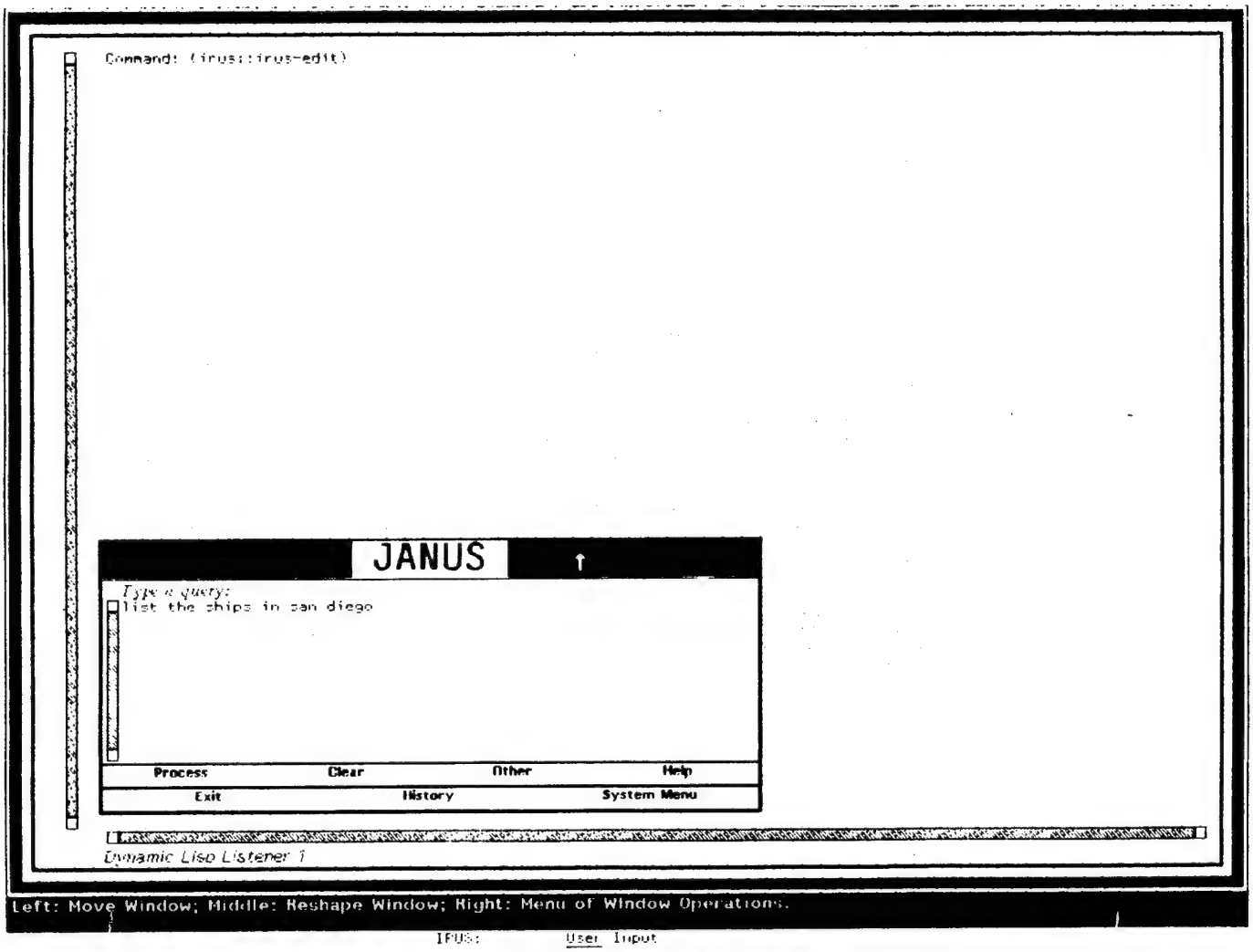


Figure 42: Move mouse to top of window

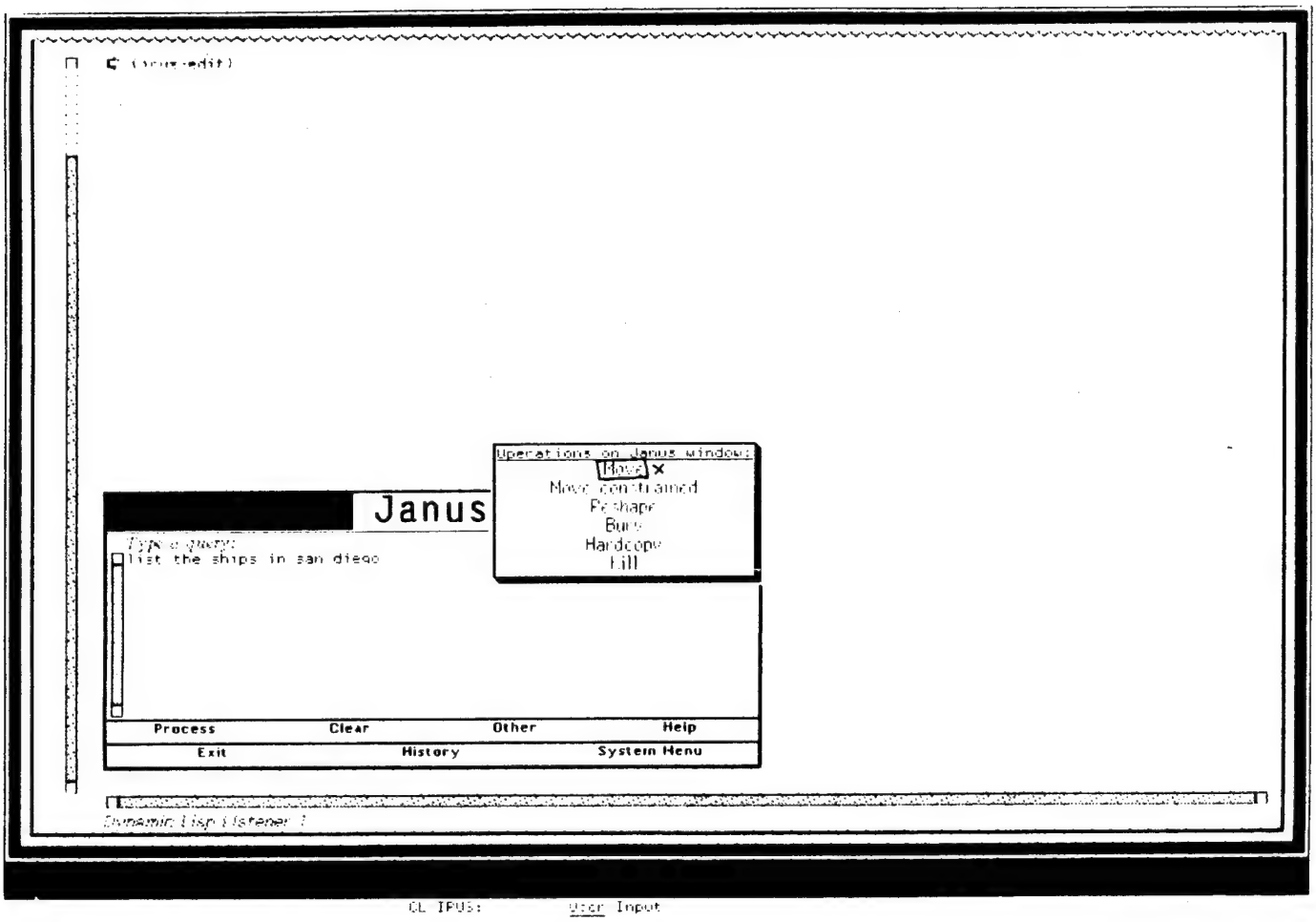


Figure 43: Menu of window operations

To move the window, click left (or on the **Move** menu selection), then move the window outline to the desired position and click the mouse (see figure 44).

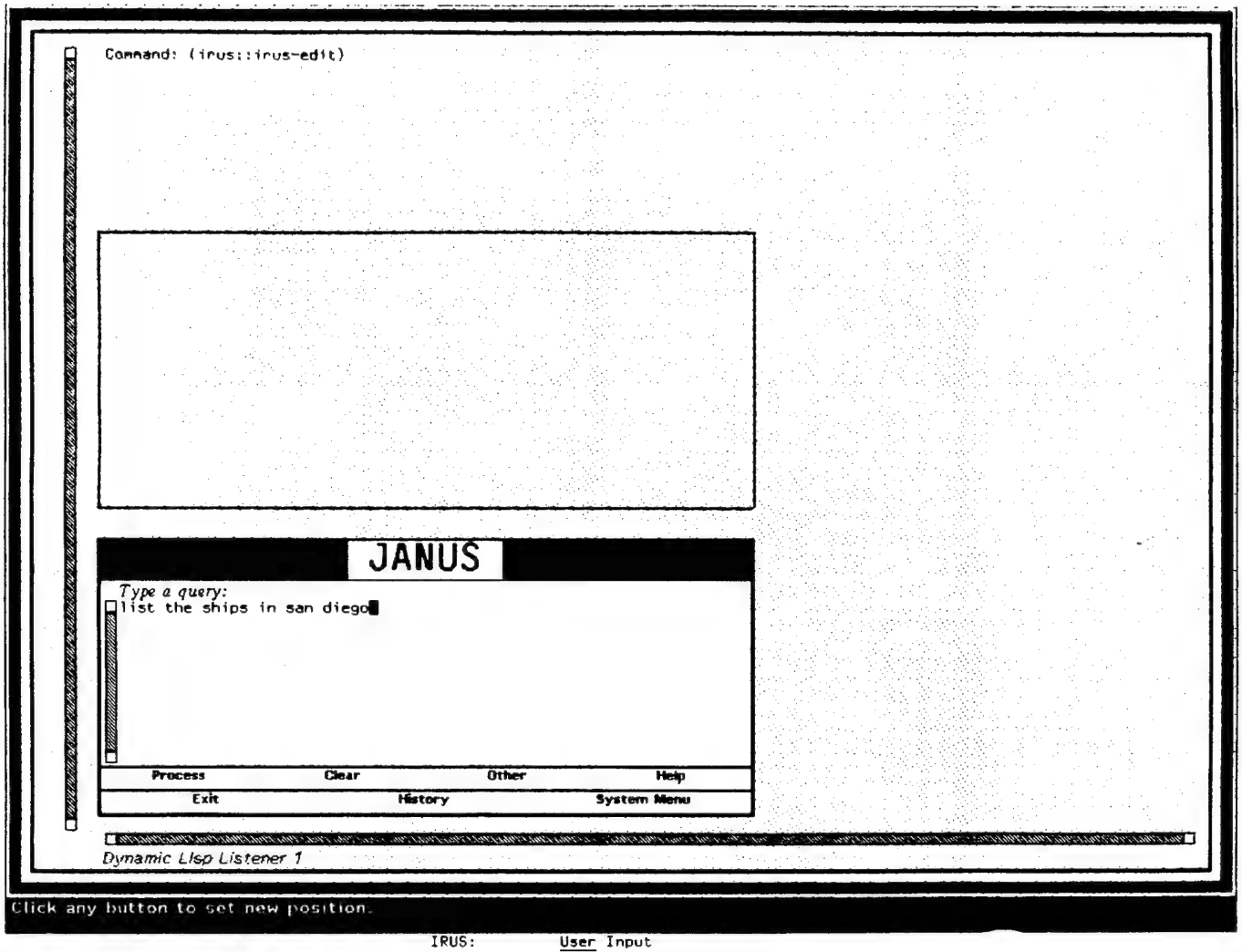


Figure 44: Move window to desired position

To reshape the window, click middle (or on the **Reshape** menu selection), then move the arrow to the edge or corner you wish to move. Hold down the left button while you move the edge or corner to the desired position.

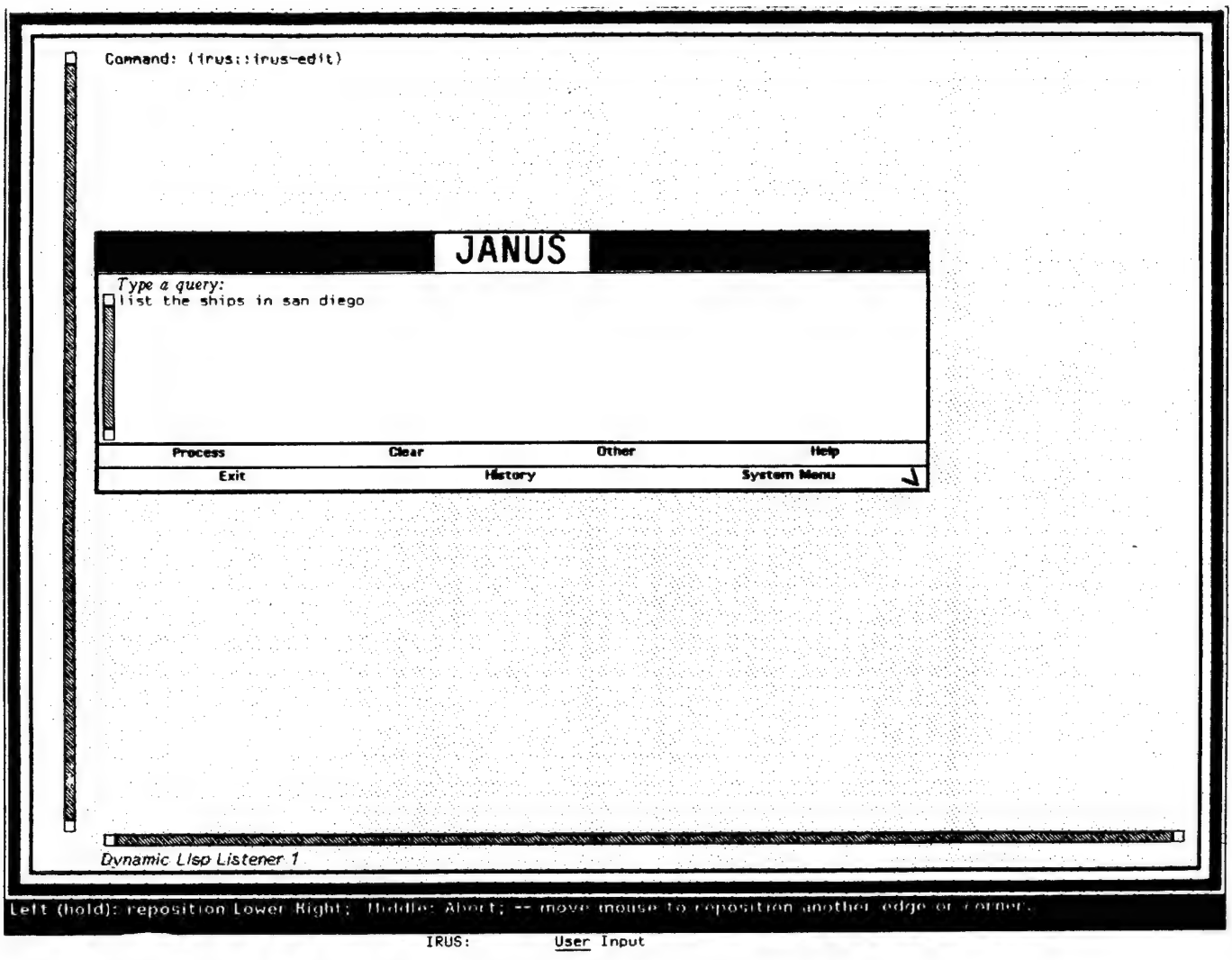
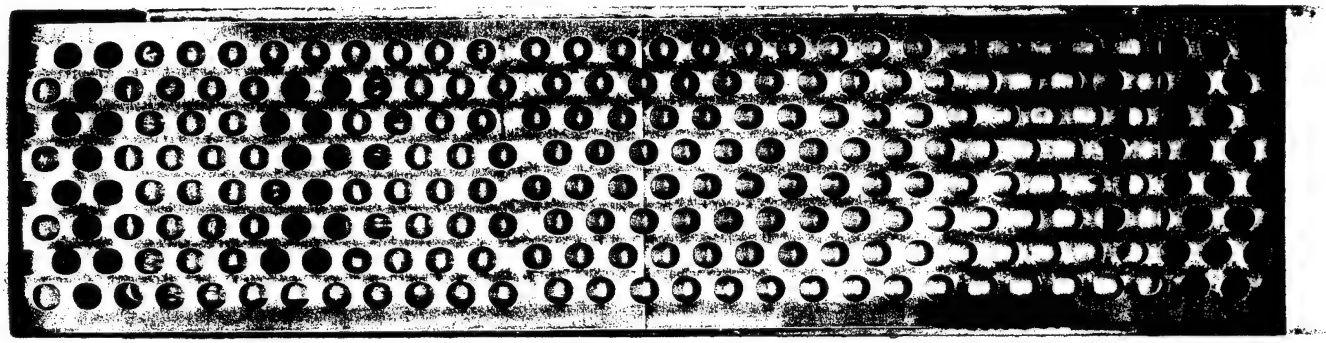
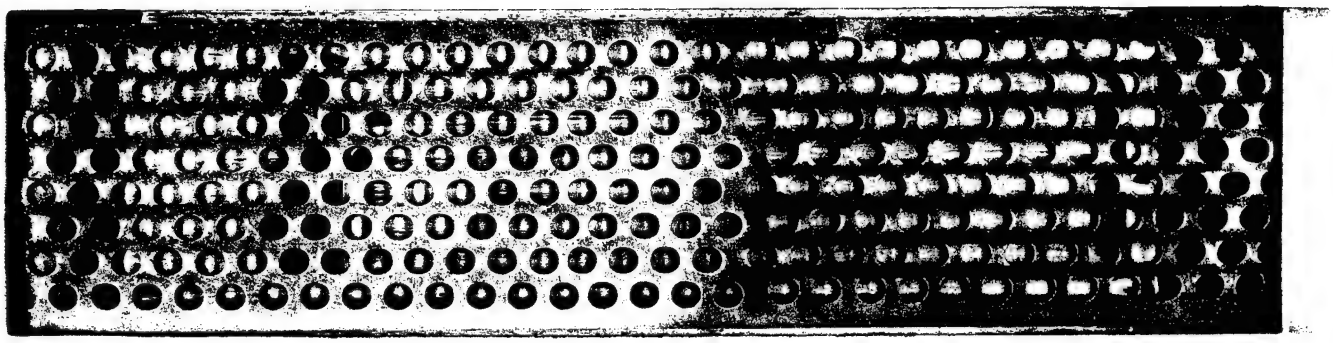


Figure 45: Move arrow to edge or corner (see lower right)





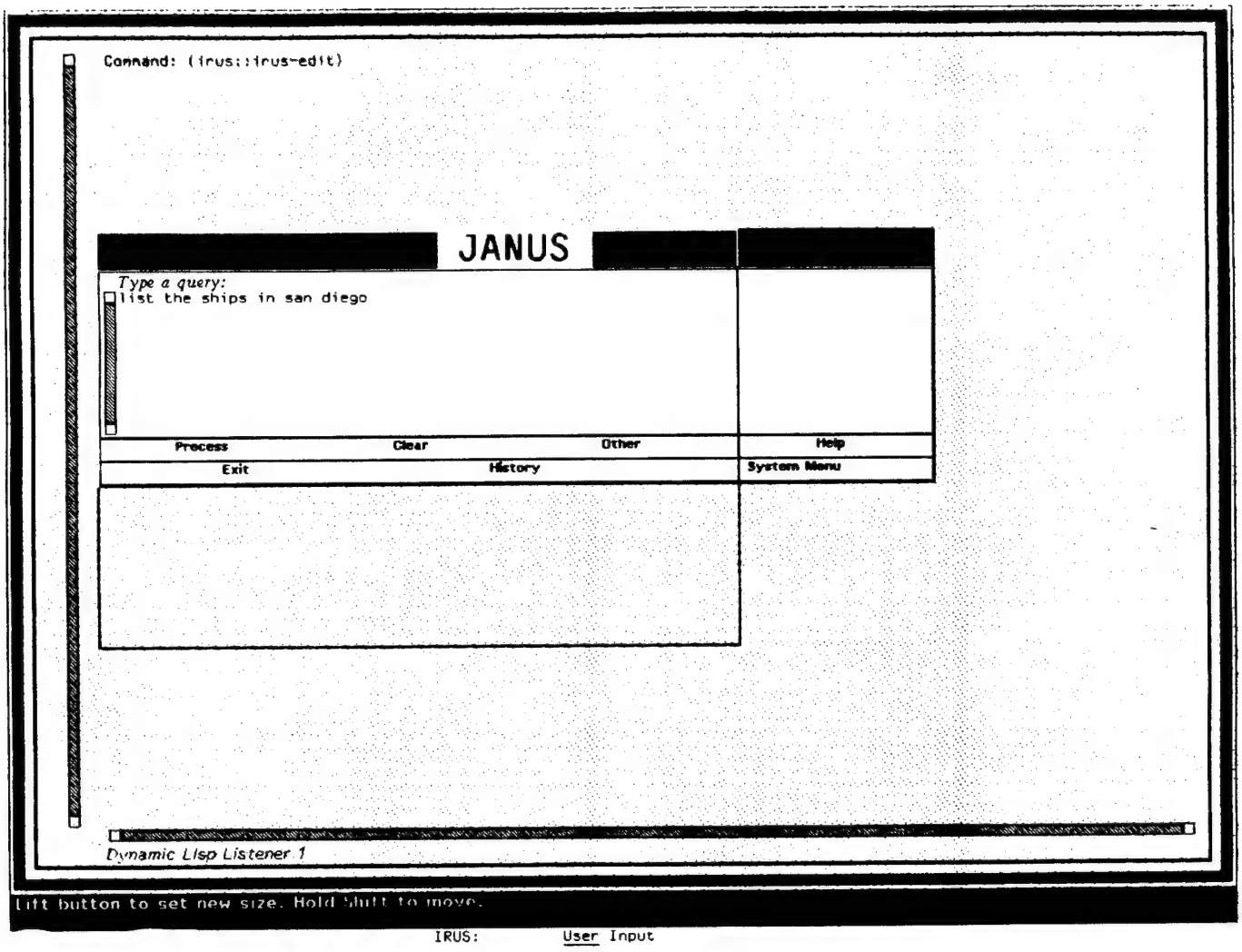


Figure 46: Hold button down and move mouse

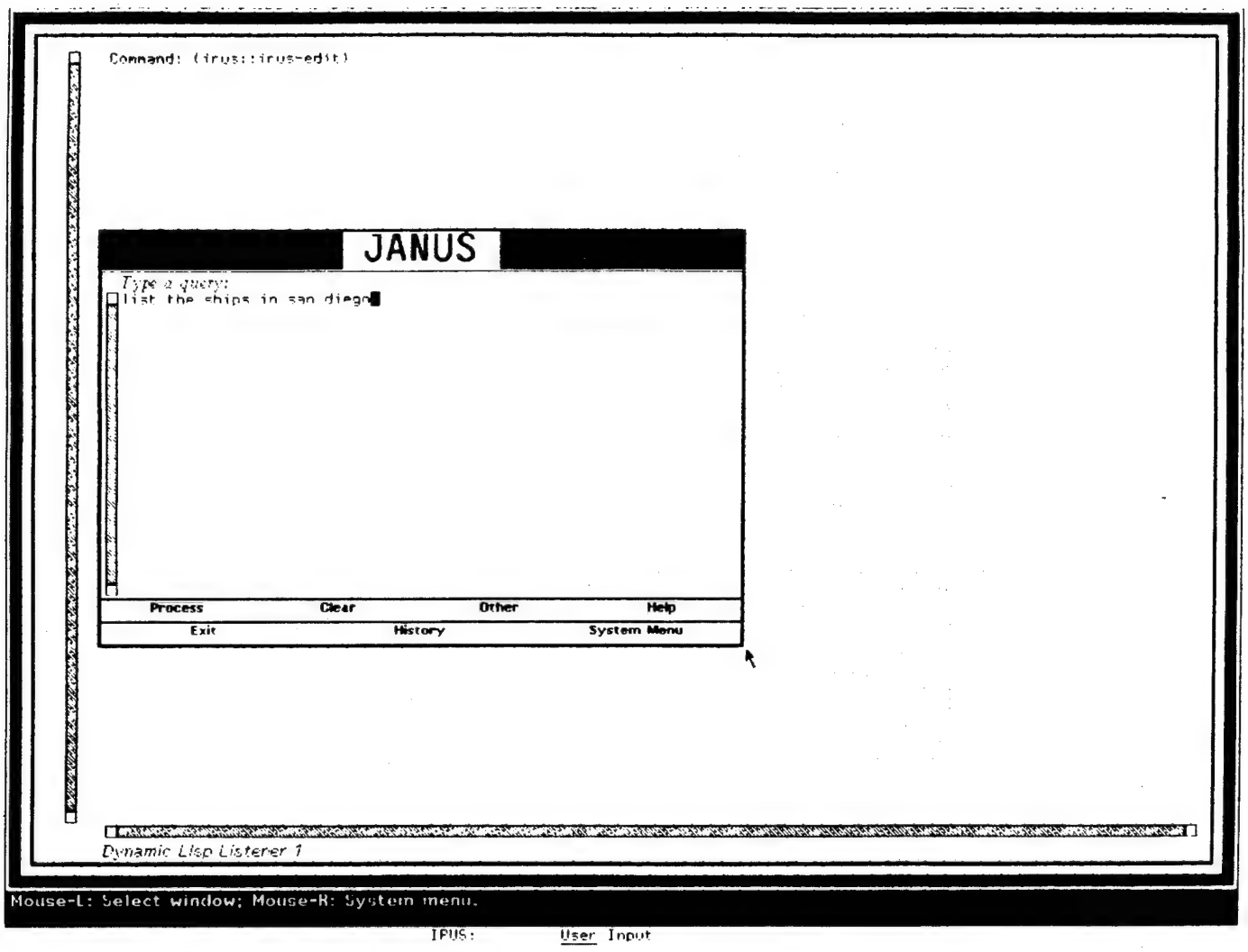


Figure 47: Release button for new position

5 Summary

The end-user's and developer's configurations of the IRUS interface illustrated here have several important features. A full editor for Lisp expressions or text is provided to aid in the composition or revision of input to IRUS; feedback on stages of processing is provided as processing proceeds; a history mechanism and facilities for incorporating help mechanisms or command composition aids is included; and the capability for setting breakpoints in IRUS processing and for examining and editing intermediate structures resulting from IRUS processing is available.

These facilities have been implemented in a flexible and modular interface system that can be adapted to a variety of applications and development needs.

Appendix A
IRUS Interface Documentation

1. To select an IRUS frame, type <SELECT> J. To create a new frame, type <SELECT> CONTROL-J.
2. To process a query (or set of queries), compose queries in the editor pane and click on PROCESS command or hit the <END> key. (See next section for other available commands.) Queries are separated in the editor pane with two carriage returns. (Actually, you can separate them by something else if you rebind the variable IRUS-I:*QUERY-DELIMITER*, which is currently bound to two carriage returns.)
3. To change configurations, click on the CONFIGURE command. Current configurations are: User, Developer, IRACQ, and Dictionary.
4. To get an IRUS pop-up frame, call the function IRUS-EDIT. (See description on page 51.)
5. Commands in editor command menu (developer configuration):
    - EDIT: get into the editor (equivalent to clicking the mouse in the editor pane.)
    - PROCESS: clicking left processes all the queries that are in the editor (equivalent to hitting the <END> key); clicking right pops up a menu allowing you to process a subset of the queries. A subset is designated by marking queries. Currently, queries are marked by preceding them with an *.
    - INTERRUPT: get out of the editor pane and into the Lisp pane (equivalent to clicking in the Lisp pane).
    - CLEAR: clear the editor.
    - SET MODE: set the editor mode to ENGLISH, PARSE-TREE, WML, REQUEST, or EPLAN. In ENGLISH mode, ZMACS auto-fill text mode is set. In the other modes, ZMACS Lisp mode is set.
    - OTHER: pops up a menu of other commands: WRITE, INSERT, ABBREV, COMMENT, and SET PROCESSING FLAGS.
        - WRITE: write queries that are in the editor to a file. You will be prompted for a file name.
        - INSERT: insert queries from a file into the editor. You will be prompted for a file name.
        - ABBREV: pops up menu of abbreviation commands, e.g., turn on ZMACS word abbreviation mode, make new word abbreviation, list word abbreviations.
        - COMMENT: save a comment in the comments file. User is given the option of also saving query objects in the file.
        - SET PROCESSING FLAGS: this calls the function SET-FLAGS which is described below.
    - HELP: pops up help menu. (This menu currently is under development.)
6. Developer's second command menu:
    - This menu appears in the developer's configuration and allows incremental processing of queries. The query can be processed to SYNTAX (syntax only, no semantics), PARSE-TREE, WML, REQUEST, EPLAN, or RESPONSE (from the database).
7. Commands in editor command menu (user configuration):

- The user's editor command menu contains a subset of the commands available for the developer: EDIT, PROCESS, INTERRUPT, CLEAR, OTHER, and HELP.

8. IRACQ interface:

   - ADD IRULE: add an IRule. You will be asked several questions about the IRule to be added.
   - EDIT IRULE: you will be prompted for an IRule name, and an editor pane containing the IRule will appear. An editor command pane will appear that contains the following commands: SAVE IRULE, INTERRUPT, ABORT, and HELP. (Note: when an IRule is edited, a query object is not created. The s-expression is put into the editor, and the IRule name is bound to IRUS:*IRULE-NAME*. The edited IRule is bound to IRUS:EDITED-IRULE.)
   - PRINT IRULE: you will be prompted for an IRule name, and the IRule will be printed out on the Lisp pane.
   - DELETE IRULE: you will be prompted for an IRule name, and the IRule will be deleted.
   - SAVE IRULES: save all IRules that have been created or edited.

9. When a query is processed from the editor pane, a query object is created that contains instance variables for each of the stages in the processing. These instance variables are also objects. A query that starts from English, for example, contains a parse tree object whose value is the s-expression that is the result of parsing the English.

   The following commands are available for operating on query objects (i.e., are available from a menu that appears when you click on a query object):

   - DESCRIBE: list the parts of the query object (in the Lisp pane).
   - EDIT: edit this query object. (Anything already in the editor will be erased.)
   - ADD: add this query object to the editor. (Anything already in the editor will remain in the editor.) This command allows you to yank several queries into the editor to be processed at the same time.
   - REPEAT: repeat this query. Query is processed to the stage currently indicated by the developers' "incremental" command menu.
   - INSPECT: call the Lisp system function INSPECT on this object.
   - SET-@: set the value of the global variable IRUS:@ to this object.

10. The following commands are available for operating on objects that are parts of query objects:

    - VIEW: display the value of this object.
    - EDIT: edit this object. (Same as EDIT for query object.)
    - ADD: add this object to the editor. (Same as ADD for query object.)
    - SET-@ : (Same as for query object.)

11. Useful Functions

    (Note: functions are in IRUS package unless otherwise specified.)

    - The following functions are available for getting the VALUES of the query parts (each takes a query object as an argument):

      QUERY-STRING, PARSE-TREE, WML, PARAPHRASE, REQUEST, EPLAN, RESPONSE, CHART, SYNTAX, IRUS-OUTPUT.

      These functions are available for getting the PARTS of the query:

      QUERY-STRING-OBJECT, PARSE-TREE-OBJECT, WML-OBJECT, PARAPHRASE-

OBJECT, REQUEST-OBJECT, EPLAN-OBJECT, RESPONSE-OBJECT, CHART-OBJECT, SYNTAX-OBJECT, IRUS-OUTPUT-OBJECT.

- IRUS-EDIT (&optional (make-new-frame nil)): *function*

This function creates a pop-up IRUS frame that contains only an editor pane and an editor command menu. The size and location of the frame are specified by the variable IRUS-I:*DEFAULT-POPUP-EDGES*. (See description on page 52.) Currently, the command menu is the same as in the user configuration, so the incremental processing stages cannot be set. (The query will be processed from English to database response; currently the response is a dummy response.) A HISTORY command has been added to create and pop-up the history frame. The size and location of the history frame are specified by the variable IRUS-I:*DEFAULT-POPUP-HISTORY-EDGES*. (If a previous pop-up history frame exists, selecting HISTORY from the menu will display the previous frame rather than create a new one.) The pop-up IRUS frame and the pop-up history frame disappear when the INTERRUPT is selected from the editor command menu, and a list of all query objects processed is returned. Calling IRUS-EDIT again selects the previous pop-up IRUS frame, or if called with an argument of T, creates and selects a new pop-up IRUS frame.

Either of these pop-up frames may be moved easily by positioning the mouse over the square in the lower right corner of the frame, holding down a mouse button, and dragging the mouse. The frames may be resized easily by positioning the mouse over the title line at the top of the frame, holding down a mouse button, and dragging the mouse.

- QDESCRIBE (object): *function*

This function partially describes an object; i.e., it only prints out relevant instance variables. It does not print out pointers to parent object, scroll-parse-item, and other things that probably are only important to someone working on the interface. (Calling DESCRIBE on the object will print out all the instance variables.) QDESCRIBE is what is called when you click on a query object in the history pane.

- QEDIT (thing): *function*

This function puts something into the editor pane. The "something" can be a query object, any of a query objects' parts (e.g. WML object), a string, or any s-expression. If QEDIT is called with a query object or query-part object, the editor mode will be set automatically for you; otherwise you must set it.

- LAST-QUERY (&optional (frame IRUS-I:*FRAME*)): *function*

This function returns the most recently processed query object. The optional argument allows you to get the current query for the pop-up frame (which is bound to IRUS-I:*POPUP-FRAME*). (Note: the fullsize frame is always bound to IRUS-I:*TOPLEVEL-FRAME*; IRUS-I:*FRAME* is bound to the IRUS frame that is currently selected -- pop-up or fullsize.)

- PREVIOUS-QUERIES (&optional (frame IRUS-I:*FRAME*)): *function*

This function returns a list of the previously processed query objects. The optional argument allows you to get the current query for the pop-up frame (which is bound to IRUS-I:*POPUP-FRAME*). (Note: the fullsize frame is always bound to IRUS-I:*TOPLEVEL-FRAME*; IRUS-I:*FRAME* is bound to the IRUS frame that currently is selected -- pop-up or fullsize.)

- CURRENT-QUERY: *function*

This function returns the query object currently being processed.

- CURRENT-QUERIES: *function*

This function returns a list of the query objects currently being processed.

- QUERIES (&optional (frame IRUS-I:*FRAME*)): *function*

This function returns a list of all the queries that have been processed in the frame. The optional argument allows you to get the queries from the pop-up frame.

- GET-QUERY (n &optional (frame IRUS-I:*FRAME*)): *function*

  This functions returns the Nth query (assuming that query numbering begins with 1). The optional argument allows you to get the current query for the pop-up frame (which is bound to IRUS-I:*POPUP-FRAME*). (Note: the fullsize frame is always bound to IRUS-I:*TOPLEVEL-FRAME*; IRUS-I:*FRAME* is bound to the IRUS frame that is currently selected -- pop-up or fullsize.)

- IRUS-I:CLEAR-EDITOR: *function*

  Clears the editor, if you are in it.

- IRUS-I:GET-EDITOR-STRING (&key common): *function*

  Returns the string in the editor. With the keyword "common", returns a Common Lisp string.

- IRUS-I:SET-EDITOR-STRING (string): *function*

  Sets the string in the editor to be the argument string.

- IRUS-I:ADD-STRING-TO-EDITOR (string): *function*

  Adds string to editor at current cursor position.

- IRUS-I:ADD-TO-EDITOR (object): *function*

  Adds string associated with object to editor at current cursor position.

- IRUS-I:EXPOSE-HISTORY: *function*

  Pops up the history pane.

- IRUS-I:HIDE-HISTORY: *function*

  Causes the pop-up history pane to disappear.

- IRUS-I:ERASE-HISTORY: *function*

  Clears the history pane and its list of all queries that have been processed.

12. IRUS-DRIBBLE (filename): *function*

    Causes everything displayed on the Lisp pane also to be recorded in a file. Calling the function again turns off the dribbling.

13. SET-FLAGS: *function*

    Pops up a menu of query processing flags. Examples of the flags you can set are: turn on paraphrasing, clear Editor pane after processing, clear Lisp pane before processing, ignore errors when processing, save debug output in dribble file when dribbling, deexpose pop-up frame after processing queries.

14. Global Variables

    - IRUS-I:*FRAME*: *variable*

      The currently selected IRUS frame (either pop-up or fullsize).

    - IRUS-I:*TOPLEVEL-FRAME*, IRUS-I:*POPUP-FRAME*: *variable*

      The fullsize IRUS frame is always bound to IRUS-I:*TOPLEVEL-FRAME*. The pop-up frame is always bound to IRUS-I:*POPUP-FRAME*.

    - IRUS-I:*DEFAULT-POPUP-EDGES*: *variable*

This variable is the default screen location for the IRUS pop-up frame. It is a list of the left, top, right, and bottom pixel locations of the edges of a rectangle. Current default is '(114 424 1031 741).

- IRUS-I:*DEFAULT-POPUP-HISTORY-EDGES*: *variable*

This variable is the default screen location for the IRUS pop-up history frame. It is a list of the left, top, right, and bottom pixel locations of the edges of a rectangle. Current default is '(164 53 1121 364).

- IRUS-I:*PROMPT*: *variable*

The Lisp pane contains a prompt that you can change by modifying the variable IRUS-I:*PROMPT*.

- IRUS-I:*QUERY-FILE*, IRUS-I:*QUERY-DIRECTORY*: *variable*

These variables specify the default file name and directory for writing (inserting) queries from (into) the editor pane.

- IRUS-I:*PRINT-STARTING-OBJECT*, IRUS-I:*PRINT-STARTING-OBJECT-NUMBER*, IRUS-I:*PRINT-ABBREV-STARTING-OBJECT*:

variable These variables control what is printed on the Lisp pane after a query has been processed. (Note: They probably will become unnecessary as the system is used and a preferred printed representation becomes apparent.) The default is for the first two to be NIL and the last one to be T. In this case, a typical output will be:

WML for "what is the Frederick's location and ...": <the WML>.

The starting object, in this case the English sentence "what is the Frederick's location and readiness", is printed in abbreviated form; i.e., only as much of it as will fit on one line is printed. To have the entire sentence printed out, set IRUS-I:*PRINT-STARTING-OBJECT* to T. To have the query number printed out (e.g. WML for Query 2), set IRUS-I:*PRINT-STARTING-OBJECT-NUMBER* to T. (Note: The printing function examines the variables in the order listed above; so if more than one is T, the first one in the ordering will determine what is printed.)

References

Ayuso, D. M., Shaked, V., and Weischedel, R. M. An Environment for Acquiring Semantic Information. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pages 32-40. ACL, 1987.

Ayuso, D., Donlon, G., MacLaughlin, D., Ramshaw, L., Resnik, P., Shaked, V., and Weischedel, R. A Guide to Irus-II Application Development. BBN Report 7144. BBN Systems and Technologies Corporation, 1989.